# Unity Pro
## Operating Modes

12/2015

33003101.17

www.schneider-electric.com

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2015 Schneider Electric. All rights reserved.

# Table of Contents


| | | |
|---|---|---|
| | **Safety Information** | **29** |
| | **About the Book** | **31** |
| **Part I** | **General** | **33** |
| **Chapter 1** | **General Presentation of Unity Pro** | **35** |
| | Capabilities of Unity Pro | **36** |
| | User Interface | **40** |
| **Chapter 2** | **Programming Language Editors** | **43** |
| | Function Block Diagram FBD | **44** |
| | Ladder Diagram (LD) Language | **46** |
| | General Information about SFC Sequence Language | **48** |
| | Instruction List IL | **52** |
| | Structured Text ST | **53** |
| **Chapter 3** | **Connecting a terminal to a PLC** | **55** |
| | Methodology for connecting from a PC to a PLC | **55** |
| **Chapter 4** | **Global project management** | **57** |
| 4.1 | Access security management | **58** |
| | Unity Pro access security management | **59** |
| | Security editor | **61** |
| | User Information | **62** |
| | User Functions | **64** |
| | User Profiles | **67** |
| | Policies | **72** |
| | Recovery procedure with an error | **74** |
| | Access rights | **75** |
| 4.2 | Project management | **80** |
| | Access to Unity Pro Software | **81** |
| | Services in Offline Mode | **83** |
| | Services in Online Mode | **92** |
| | Allowed Online Modifications | **94** |
| | Connection / disconnection | **99** |
| | Accessing a PLC via a Network | **101** |
| | Overview of Types of Connection Possible With Schneider PLCs | **107** |
| | Accessing communication parameters | **111** |
| | Project comparison | **114** |
| | Conversion of Legacy Applications/Projects | **115** |

Importing an LL984 Application . . . . . 117
Transferring the project between the terminal and the PLC. . . . . 125
Update of Upload Information . . . . . 129
Sending a command to the PLC . . . . . 131
Project Backup Management for Modicon M340 . . . . . 132
Project backup for Premium. . . . . 134
Save/Restore Data Between a File and the PLC . . . . . 136
Compatibility Rules for Restore Using a *.DTX File . . . . . 140
Transferring current values . . . . . 146
Memory usage . . . . . 147
Function Pack for Memory . . . . . 152
Memory Card Access for Modicon M340 . . . . . 155
FTP and Memory Card for Modicon M340. . . . . 156
Project Management with DTMs . . . . . 158
**Chapter 5 Project browser . . . . . 161**
5.1 Introduction to the project browser. . . . . 162
Introduction to the project browser. . . . . 163
Structural view and functional view of the project . . . . . 165
The Zoom In and Zoom Out commands . . . . . 167
The commands Contract all and Expand all . . . . . 170
The Goto command . . . . . 174
User Directory . . . . . 177
Project Properties. . . . . 179
Application Protection . . . . . 180
Firmware Protection . . . . . 184
Section and Subroutine Protection. . . . . 186
Data Storage Protection. . . . . 188
Loss of Password. . . . . 190
5.2 Introduction to the various directories of the project browser. . . . . 193
Station Directory. . . . . 194
Configuration Directory . . . . . 196
Derived data types (DDT) directory . . . . . 198
Derived FB (DFB) types directory . . . . . 199
Variables Directory. . . . . 200
Motion Directory . . . . . 202
Communication Directory. . . . . 204
Program Directory . . . . . 206

Animation Tables Directory . . . . . 209
Operator Screens Directory . . . . . 211
Documentation Directory . . . . . 213
Conversion Report Directory . . . . . 214
5.3 Summaries of the utilities associated with the structure view and function view . . . . . 215
Summary of the services associated with the structural view . . . . . 215
**Chapter 6 Functional modules . . . . . 221**
6.1 Introduction to the functional view . . . . . 222
Functional modules . . . . . 223
Station directory of the functional view . . . . . 225
Functional Module Directory . . . . . 226
Functional Module Program Directory . . . . . 228
Functional Module Animation Tables Directory . . . . . 229
Functional Module Operator Screens Directory . . . . . 230
Summary of the services associated with the functional view . . . . . 231
6.2 The functional modules and their associated utilities . . . . . 234
Properties of a functional module . . . . . 235
Creating a functional module . . . . . 236
Programming a functional module . . . . . 237
Protecting program elements of the functional module . . . . . 239
Debugging a functional module . . . . . 240
Detaching/Deleting a functional module . . . . . 241
Functional module export . . . . . 245
Functional module import . . . . . 246
Creating, Deleting, Locating, Dragging-and-Dropping an Animation Table in a Functional Module . . . . . 247
Creating, Deleting, Locating and Dragging-and-Dropping an Operator Screen in a Functional Module . . . . . 250
**Chapter 7 FDT Container . . . . . 253**
7.1 FDT Container . . . . . 254
Introduction to the Unity Pro FDT Container . . . . . 255
FDT Container User Rights . . . . . 256
7.2 DTM Hardware Catalog . . . . . 257
Description of DTM Hardware Catalog . . . . . 257
7.3 DTM Browser . . . . . 262
DTM Browser . . . . . 263
Ethernet Ready Equipment . . . . . 268
Ready, Built, Connected and Installed DTMs . . . . . 270

DTM Browser Contextual Menus . . . . . 273
Field Bus Discovery Service . . . . . 281
Store Device Configurations . . . . . 284
Renaming DTMs . . . . . 285
Profibus DTMs . . . . . 287
7.4 DTM Properties Tabs . . . . . 288
DTM Properties Dialog . . . . . 288
**Chapter 8 Type Library Manager . . . . . 293**
Libset (Set of Libraries) . . . . . 294
Overview of the Type Library Manager . . . . . 297
Type Library Manager . . . . . 298
Description of the Type Library Manager . . . . . 301
Loading an object from the library into the project . . . . . 307
Transferring an object from the project into the library . . . . . 310
Creating a new library and family . . . . . 312
Creating an installable family . . . . . 314
Updating a family of a library . . . . . 315
Consistency Check Assistant . . . . . 317
Library Version Management . . . . . 320
Using the type library of an older Unity Pro Version . . . . . 324
**Part II Data Editor . . . . . 325**
**Chapter 9 Description of the Data Editor . . . . . 327**
Data Editor Access . . . . . 328
Description of Data Editor tabs . . . . . 330
**Chapter 10 Creating derived data types (DDT) . . . . . 333**
Access to derived data types (DDT) . . . . . 334
Creation of a derived data type (DDT) . . . . . 335
Administration of derived data types and their elements . . . . . 346
Creating Nested Derived Data Types . . . . . 348
Save Derived Data Types (DDT) . . . . . 355
**Chapter 11 Creating data types of user function blocks (DFB) . . 357**
Creation and Management of Data Types and Function Blocks (DFB) 357
**Chapter 12 Managing of instances of data belonging to the family of function blocks (EF) . . . . . 359**
Creating / Deleting a variable instance of function blocks (EFB/DFB) . 360
Displaying function block variable instances in the editor . . . . . 363
Changing the attribute for variable instances in a function block . . . . . 366

**Chapter 13 Managing variable instances belonging to the EDT/DDT/IODDT/Device DDT families . . . . . 369**
Creation of EDT type variable instances . . . . . 370
Creation of variable DDT type instances . . . . . 375
Creating a type IODDT variable instance . . . . . 382
Creating a Type Device DDT Variable instance . . . . . 385
Managing an IODDT, DDT or an Array type instance . . . . . 387
Creation of multiple variable instances . . . . . 393
Displaying the variable instances in the editor . . . . . 395
Changing attributes of variable instances . . . . . 399
Creation of a Reference Type Instance . . . . . 402
**Chapter 14 General Functions . . . . . 403**
Configuring the Data Editor columns . . . . . 404
Description of the Data Editor: Variable type selection . . . . . 408
Filtering of Data . . . . . 412
Exporting a subset of variables . . . . . 416
Analyze DDT and DFB Data Types . . . . . 417
Purge of Unused Instances . . . . . 418
Local printing of data . . . . . 420
Save the context of the Data Editor . . . . . 421
**Part III Communication . . . . . 423**
**Chapter 15 Presentation of the communication editors . . . . . 425**
Presentation of the communication editors of the project browser . . . 425
**Chapter 16 Network Configuration . . . . . 427**
Network Configuration Principle Using Unity Pro . . . . . 428
Creating a Logic Network . . . . . 429
Configuring a Logic Network . . . . . 431
Associating a Logic Network with Network Hardware . . . . . 432
**Chapter 17 Configuration of X-Way Routing Premium Stations . . 435**
Configuration . . . . . 436
Configuration of Multi-Network Services . . . . . 437
Configuring an X-Way Router Module . . . . . 439
Examples of X-Way Routing Stations . . . . . 443
Examples of Partial Routing . . . . . 447

**Part IV Programming . . . . . 451**
**Chapter 18 Accessing Tasks and Sections . . . . . 453**
18.1 Access to Tasks . . . . . 454
Create and Configuration of a Task . . . . . 455
Displaying and Modifying Task Properties . . . . . 456
18.2 Accessing Sections . . . . . 457
Creating an FBD, LD, IL, or ST Section . . . . . 458
Properties Dialog Box for FBD, LD, IL, or ST Sections . . . . . 461
Creating SFC sections . . . . . 467
Properties Dialog Box for SFC Sections . . . . . 470
Creating action, transition and macro sections . . . . . 477
Action, transition and macro sections properties dialog box . . . . . 478
Create a Sub-Program Section (SR) . . . . . 481
Property Dialog of the Sub-Program Sections . . . . . 483
Creating a Timer Event Section . . . . . 486
Properties dialog box for Timer Event Sections . . . . . 489
Create an I/O Event Section . . . . . 493
Property Dialog of I/O Event Sections . . . . . 496
Opening a Section . . . . . 500
Deleting sections . . . . . 501
Execution order of sections . . . . . 502
Creating LL984 Segments and Networks . . . . . 503
Modifying LL984 Segment Properties . . . . . 508
Modifying LL984 Network Properties . . . . . 510
Scheduling LL984 Segments and Network . . . . . 512
Deleting, Cutting, Copying and Pasting LL984 Segments and Network 513
**Chapter 19 Unity Pro Software Options . . . . . 515**
19.1 Project Settings . . . . . 516
General Description . . . . . 517
General Project Settings . . . . . 519
Variables . . . . . 531
Program . . . . . 534
Maintain Output Links on Disabled EF (EN=0) . . . . . 542
Configuration Settings for Operator Screens . . . . . 550

19.2 Options . . . . . 554
General Description . . . . . 555
General Options . . . . . 557
Data and Languages . . . . . 559
Connection Options . . . . . 561
The Operator Screens Option . . . . . 563
Converters . . . . . 565
19.3 Customize Dialog Box . . . . . 571
Customize Dialog Box . . . . . 571
**Chapter 20 Common Function for graphical language . . . . . 575**
Using Bookmarks in Graphical Languages Sections . . . . . 576
Display grid in a FBD/LD/SFC section . . . . . 581
Zoom in a FBD/LD/SFC section . . . . . 582
Tracking links in a FBD/LD/SFC section . . . . . 585
Print Current Section in a FBD/LD/SFC section . . . . . 586
Undo and redo changes in a FBD/LD section . . . . . 587
Using connectors in a FBD/LD section . . . . . 588
Inspect windows in a FBD/LD section . . . . . 593
Tooltips for Variables in a FBD/LD/SFC Section . . . . . 597
Replace Variables in a FBD/LD Section . . . . . 599
Initialize search in a FBD/LD section . . . . . 603
Initialize Animation Table in a FBD/LD section . . . . . 604
Go to in a FBD/LD section . . . . . 605
Go to dialog box in a FBD/LD section . . . . . 607
**Chapter 21 FBD Editor . . . . . 611**
21.1 Creating a program with FBD programming language . . . . . 612
Structure of an FBD program (function block language) . . . . . 613
Creating an FBD Program . . . . . 615
Syntax and Semantics Check during programming . . . . . 616
Navigating with the keyboard . . . . . 617
Selecting objects . . . . . 618
Deleting, cutting, copying, pasting and moving objects . . . . . 621
Inserting Rows and Columns . . . . . 625
Deleting Rows and Columns . . . . . 626
Displaying the properties . . . . . 627
Refining DFBs and subroutines . . . . . 628

21.2 Editing FFBs . . . 630
General information about calling an FFB . . . 631
Calling an FFB via the FFB Input Assistant . . . 633
Calling an FFB via the data selection . . . 637
Assign actual parameters . . . 642
Using Public Variables . . . 654
Expanding Functions . . . 656
Negating FFB Pins . . . 657
EN and ENO Show and Hide . . . 658
Modifying the execution order . . . 659
Replacing an FFB . . . 663
Properties dialog box for elementary functions, procedures and sub-routine blocks . . . 665
Properties dialog box for elementary and derived function blocks (FBs) 667
21.3 Returning from a Subroutine or DFB . . . 670
Returning from a Subroutine or DFB . . . 671
Return object properties dialog box . . . 673
21.4 Calling a Subroutine . . . 674
Calling a subroutine . . . 674
21.5 Jumps within the Current Section . . . 678
Jump within the current section . . . 679
Jump object properties dialog box . . . 683
21.6 Definition of the Jump Target (Jump Labels) . . . 685
Definitions of Jump Labels . . . 686
Jump label objects properties dialog box . . . 688
21.7 Editing Links . . . 690
Links . . . 691
Placing a Link . . . 693
Editing Links . . . 696
21.8 Entering Comments . . . 702
Entering comments . . . 702
21.9 Online functions . . . 704
Online functions . . . 704
21.10 FBD Reference Data Type . . . 705
Reference Data Type in FBD . . . 705
21.11 Export/Import . . . 706
Export/Import FBD Sections . . . 706

21.12 Customize FBD Editor Colors . . . . . 707
Customize FBD Editor Colors . . . . . 707
**Chapter 22 LD Editor . . . . . 709**
22.1 Creating a program with LD programming language . . . . . 710
Structure of an LD program (ladder diagram) . . . . . 711
Creating an LD program . . . . . 713
Syntax and Semantics Check during programming . . . . . 717
View for variables . . . . . 719
Navigating with the Keyboard . . . . . 724
Selecting objects . . . . . 727
Deleting, cutting, copying, pasting and moving objects . . . . . 730
Displaying the properties . . . . . 734
Refining DFBs and subroutines . . . . . 735
Inserting and Deleting Rows . . . . . 737
22.2 Editing Contacts . . . . . 738
Select contacts . . . . . 739
Placing contacts . . . . . 741
Property dialog box for contacts . . . . . 745
22.3 Editing Coils . . . . . 748
Selection of coils . . . . . 749
Placing coils . . . . . 752
Property dialog box for coils . . . . . 758
22.4 Editing compare blocks . . . . . 761
Placing compare blocks . . . . . 762
Compare block properties dialog box . . . . . 766
22.5 Editing operation blocks . . . . . 768
Placing Operate Blocks . . . . . 769
Operate block properties dialog box . . . . . 774
22.6 Editing FFBs . . . . . 776
General information on calling an FFB . . . . . 777
Calling an FFB via the FFB input assistant . . . . . 779
Calling an FFB via the data selection . . . . . 783
Connection of FFBs to the Left Power Rail . . . . . 788
Assigning Actual Parameters . . . . . 789
Using Public Variables . . . . . 801
Expanding Functions . . . . . 803

Negating FFB Pins . . . 805
EN and ENO Show and Hide . . . 806
Properties dialog box for elementary functions, procedures and sub-routine blocks. . . . 807
Properties dialog box for elementary and derived function blocks (FBs) 809
22.7 Returning from a Subroutine or DFB . . . 811
Returning from a subroutine or DFB . . . 812
Return object properties dialog box . . . 814
22.8 Jumps within the Current Section . . . 815
Jumps within the current section . . . 816
Jump object properties dialog box . . . 819
22.9 Definition of the Jump Target (Jump Labels) . . . 821
Definitions of jump targets (jump labels) . . . 822
Jump label properties dialog box . . . 825
22.10 Editing Links . . . 827
Select links . . . 828
Combining Links . . . 830
Placing links . . . 832
Editing links . . . 847
22.11 Entering Comments . . . 853
Entering comments . . . 853
22.12 Online functions . . . 855
Online Functions . . . 855
22.13 LD Reference Data Type . . . 856
Reference Data Type in LD . . . 856
22.14 Export/Import . . . 857
Export/Import LD Sections . . . 857
22.15 Customize LD Editor Colors . . . 858
Customize LD Editor Colors . . . 858
**Chapter 23 SFC Editor . . . 861**
23.1 Creating a program with the SFC programming language . . . 862
Structure of an SFC Program (Sequential Function Chart) . . . 863
Creating an SFC Program . . . 866
Syntax and Semantics Check during Programming . . . 867
Navigating with the keyboard . . . 869
Selecting Objects . . . 870
Undo and Redo Changes . . . 873
Deleting, cutting, copying, pasting and moving objects . . . 874

| | | |
|---|---|---|
| | Inserting Rows and Columns | 880 |
| | Deleting Rows and Columns | 882 |
| | Displaying the object properties | 884 |
| | Initialize Search | 885 |
| | Initialize Animation Table | 886 |
| | Refining steps, macro steps and transitions | 887 |
| | Go to | 889 |
| | Go to dialog | 891 |
| | List and display jumps | 895 |
| | Go to Step | 896 |
| 23.2 | General Information about Steps | 897 |
| | Selecting steps | 898 |
| | Placing steps | 900 |
| 23.3 | Editing "normal" steps | 901 |
| | Defining the properties of steps | 902 |
| | Steps properties dialog box | 909 |
| 23.4 | Editing macro steps | 915 |
| | Defining the properties of macro steps | 916 |
| | Macro Steps Properties Dialog Box | 923 |
| 23.5 | Editing input steps | 926 |
| | Defining the properties of InSteps | 927 |
| | InSteps properties dialog box | 929 |
| 23.6 | Editing Output Steps | 933 |
| | Defining the properties of OutSteps | 934 |
| | OutStep properties dialog box | 936 |
| 23.7 | Editing actions | 939 |
| | Assigning actions to a step | 940 |
| | Display modes for actions | 949 |
| 23.8 | Editing Transitions | 951 |
| | Placing transitions | 952 |
| | Defining transition properties | 954 |
| | Properties dialog box for transitions | 960 |
| 23.9 | Editing Jumps | 963 |
| | Placing jumps | 964 |
| | Defining jump properties | 966 |
| | Jump properties dialog box | 968 |

23.10 Editing Branches and Joints . . . 970
Selecting Branches and Joints . . . 971
Placing branches and joints . . . 973
Modifying the Properties of Branches and Joints . . . 974
Branches properties dialog box . . . 977
Joints properties dialog box . . . 978
23.11 Inserting Alternative Strings and Simultaneous Strings . . . 979
Selecting Alternative Sequences and Parallel Sequences . . . 980
Placing alternative sequences and parallel sequences . . . 981
23.12 Inserting Step-Transitions and Transition-Step-Sequences . . . 984
Selecting Step-Transition and Transition-Step Sequences . . . 985
Placing step-transition and transition-step sequences . . . 986
23.13 Editing Links . . . 989
Links . . . 990
Placing a Link . . . 991
Editing links . . . 995
23.14 Entering Comments . . . 1000
Entering comments . . . 1000
23.15 Online functions . . . 1002
Online functions . . . 1002
23.16 Export/Import . . . 1003
Export/Import SFC Sections . . . 1003
23.17 Customize SFC Editor Colors . . . 1004
Customize SFC Editor Colors . . . 1004
**Chapter 24 Common Function for Textual Language . . . 1007**
Selecting Text . . . 1008
Deleting, cutting, copying, pasting and moving text . . . 1010
Undoing and redoing changes . . . 1012
Match case . . . 1013
Zoom (font size) . . . 1014
Using Bookmarks . . . 1015
Go to . . . 1017
Go to dialog box . . . 1019
Searching text . . . 1022
Searching and replacing text . . . 1024
Call a subroutine . . . 1026
Refine DFBs and subroutines . . . 1028
Inspect windows . . . 1030

Enter comments . . . . . 1034
Printing the Current Section . . . . . 1036
Initialize search . . . . . 1037
Initialize Animation table . . . . . 1038
Displaying the section properties . . . . . 1039
Fullscreen . . . . . 1040
**Chapter 25 IL Editor . . . . . 1041**
Structure of an IL program . . . . . 1042
Creating an IL Program . . . . . 1044
Syntax and Semantics Check during Programming . . . . . 1045
Navigating with the keyboard . . . . . 1049
Data input . . . . . 1050
Calling an FFB . . . . . 1057
Using Public Variables . . . . . 1068
Online functions . . . . . 1070
Reference Data Type in IL . . . . . 1071
Export/Import IL Sections . . . . . 1072
**Chapter 26 ST Editor . . . . . 1073**
Structure of an ST program . . . . . 1074
Creating an ST Program . . . . . 1076
Syntax and Semantics Check during Programming . . . . . 1077
Navigating with the keyboard . . . . . 1081
Data input . . . . . 1082
Calling an FFB . . . . . 1088
Using public variables . . . . . 1099
Calling a statement . . . . . 1101
Online functions . . . . . 1107
Reference Data Type in ST . . . . . 1108
Export/Import IL Sections . . . . . 1109
**Chapter 27 LL984 Editor . . . . . 1111**
27.1 Creating a Program with LL984 Programming Language . . . . . 1112
Project Settings for LL984 Programs . . . . . 1113
Structure of an LL984 Program . . . . . 1114
Creating an LL984 Program . . . . . 1117
Syntax and Semantics Check During Programming . . . . . 1120
Network Display . . . . . 1123
Navigating with the Keyboard . . . . . 1124

Selecting Objects . . . . . 1126
Deleting, Cutting, Copying, Pasting and Moving Objects . . . . . 1128
Displaying the Properties . . . . . 1131
Inserting and Deleting Rows/Columns . . . . . 1132
27.2 Editing Contacts . . . . . 1134
Selecting Contacts . . . . . 1135
Placing Contacts . . . . . 1137
Properties Dialog for Contacts . . . . . 1141
27.3 Editing Coils . . . . . 1144
Selecting Coils . . . . . 1145
Placing Coils . . . . . 1147
Property Dialog for Coils . . . . . 1150
27.4 Editing Function Blocks . . . . . 1152
Editing Function Blocks . . . . . 1153
Using Mnemonics . . . . . 1156
27.5 Editing Links . . . . . 1157
Selecting Links . . . . . 1158
Combining Links . . . . . 1159
Placing Links . . . . . 1160
Editing Links . . . . . 1162
27.6 Editing Equation Network Blocks . . . . . 1163
Editing Equation Network Blocks . . . . . 1164
Editing the ST Assignment Expression . . . . . 1167
27.7 Assigning Variables to LL984 Objects . . . . . 1171
Assigning Variables to LL984 Objects . . . . . 1171
27.8 Calling Subroutines . . . . . 1175
Calling Subroutines . . . . . 1175
27.9 Entering Comments . . . . . 1177
Entering Comments . . . . . 1177
27.10 Customize LL984 Editor Colors . . . . . 1179
Customize LL984 Editor Colors . . . . . 1179
**Chapter 28 Programming Languages Data Selection Dialog Box** 1181
Variable selection dialog box . . . . . 1182
FFB Selection Dialog box . . . . . 1184
FFB Type Selection Dialog box . . . . . 1187

Data selection dialog box . . . . . 1189
Function Input Assistant . . . . . 1194
Data Properties . . . . . 1197
Generate variables . . . . . 1199
**Chapter 29 I/O Objects tab . . . . . 1201**
Introduction to the I/O Object Tab . . . . . 1202
I/O Objects Tab for a Processor . . . . . 1204
I/O Objects Tab for a Module . . . . . 1207
Details of Areas in the I/O Objects Tab . . . . . 1209
I/O Objects Tab for a Bus Device . . . . . 1214
How to Use the I/O Objects Tab . . . . . 1215
**Chapter 30 Memory Tab . . . . . 1219**
Memory Tab . . . . . 1219
**Chapter 31 Type Library Browser. . . . . 1225**
Access to the Type Library Browser . . . . . 1226
Description of the Type Library Browser . . . . . 1228
Configuration of Columns of the Type Library Browser . . . . . 1231
Drag & Drop of FFBs in a program section . . . . . 1233
**Chapter 32 Search/Replace Tool . . . . . 1235**
Overview of the Search / Replace tool . . . . . 1236
General Features for Search/Replace . . . . . 1238
Searching for Data . . . . . 1239
Replacing Data . . . . . 1240
Search Toolbar . . . . . 1241
Data Search Dialog . . . . . 1242
Cross-References (Search Results) . . . . . 1244
Cross-References (Replace) . . . . . 1247
Tracing Variables / Addresses . . . . . 1249
**Chapter 33 User's Function Blocks . . . . . 1251**
33.1 DFB Type . . . . . 1252
DFB Type . . . . . 1253
Creating a DFB Type . . . . . 1255
Configuration of a DFB Type . . . . . 1256
Programming a DFB type . . . . . 1259
Deleting a DFB type element . . . . . 1262
Archiving a DFB Type . . . . . 1263

How to protect a DFB type . . . . . 1264
Deleting a DFB Type . . . . . 1266
How to access the attributes of a DFB type . . . . . 1267
Authorized DFB Type Modifications . . . . . 1269
33.2 DFB Instance . . . . . 1272
DFB Instance . . . . . 1273
Creating a DFB Instance . . . . . 1274
Creating several DFB Instances . . . . . 1276
Deleting a DFB instance . . . . . 1277
Operations on DFB Instances . . . . . 1278
Modifying the Attributes of a DFB Instance . . . . . 1280
Authorized DFB Instance Modifications . . . . . 1283
33.3 How to Create Nested DFBs . . . . . 1284
How to Create Nested DFBs . . . . . 1284
**Part V Project Configuration . . . . . 1285**
**Chapter 34 Introduction to the configuration . . . . . 1287**
Proposed Services . . . . . 1288
Services offered in offline mode . . . . . 1292
Services Offered In Online Mode . . . . . 1293
Access to project configuration . . . . . 1295
**Chapter 35 Bus Editors . . . . . 1299**
35.1 Rack configuration on local bus . . . . . 1300
Organization of the Racks . . . . . 1301
How to configure the racks . . . . . 1306
35.2 Configuration of the power supply modules . . . . . 1308
How to configure the power supply modules . . . . . 1308
35.3 Configuration of the processor . . . . . 1311
Choice and replacement of the processor . . . . . 1311
35.4 Configuration of the modules in the PLC station . . . . . 1314
How to configure the modules in the PLC station . . . . . 1314
35.5 Configuration of devices on the field bus . . . . . 1319
Configuration of the CANopen Slaves for Modicon M340 . . . . . 1320
How to access and create a Fipio field bus . . . . . 1324
Creating and Accessing RIO\DIO Field Buses . . . . . 1328
How to configure devices on the RIO/DIO bus . . . . . 1334
35.6 Consumption management . . . . . 1340
Power consumption budget . . . . . 1341
Budget on the number of application-specific channels . . . . . 1344

35.7 Bus editors in online mode . . . . . 1347
Online Information . . . . . 1347
**Chapter 36 Modules editor . . . . . 1351**
36.1 Configuration of Modicon M340 processors . . . . . 1352
Configuration of Modicon M340 Processors . . . . . 1352
36.2 Configuration of Premium processors . . . . . 1363
Configuration of Premium\Atrium processors . . . . . 1364
Access to PCMCIA memory cards of Premium processors . . . . . 1370
Access to control loop configuration for Premium processors . . . . . 1372
36.3 Configuration of the Quantum processors . . . . . 1373
Configuration of Quantum processors . . . . . 1374
Configuration of Modbus ports on Quantum . . . . . 1380
36.4 Configuration of the input\output modules for a Premium\Atrium, Quantum or Modicon M340 station . . . . . 1383
Access to the Modicon, Premium or Atrium I/O module configuration editor . . . . . 1384
Access the Quantum input/output module configuration editor . . . . . 1386
**Chapter 37 General functionalities of the editor . . . . . 1389**
Analysis of the configuration . . . . . 1390
Export\Import of the configuration . . . . . 1391
Printing the configuration . . . . . 1392
**Part VI Debugging and adjustment . . . . . 1393**
**Chapter 38 Debugging the program . . . . . 1395**
38.1 Safety precautions . . . . . 1396
Modification in RUN . . . . . 1397
Switch to STOP of a PLC in Debug mode . . . . . 1398
Setting a task to STOP . . . . . 1399
Task deactivation . . . . . 1400
Deactivation of Inputs/Outputs . . . . . 1401
Use of the Step by Step mode and Breakpoint . . . . . 1402
Unwanted disconnection in Debug mode for Premium . . . . . 1403
Debug Screen Via a Fipway, Fipio or Uni-Telway Network with a PCMCIA Card . . . . . 1404
Behavior of Forced Variables between Modsoft/NxT/Concept and Unity Pro . . . . . 1405
38.2 Program Animation . . . . . 1406
Introduction to animation . . . . . 1407
Section in textual languages (Structured Text or Instruction List) . . . . 1410
Section in a Ladder Diagram language . . . . . 1411

Section in Function Block Diagram (FBD) language . . . . . 1414
Section in Sequential Function Chart language (SFC) . . . . . 1416
Display of Prepositioned steps, after an on-line modification. . . . . 1418
DFB instances (user function blocks). . . . . 1419
Representation of forced variables . . . . . 1421
38.3 Debugging in Ladder Language. . . . . 1422
Inserting a breakpoint in Ladder Diagram language (LD) . . . . . 1423
Step by step execution mode in Ladder Diagram language (LD) . . . . 1425
Inserting a Watchpoint in Ladder Diagram Language . . . . . 1428
38.4 Debugging in Textual Languages (Structured Text, Instruction List) . . 1431
Insertion of a breakpoint in textual languages (Structured Text or Instruction List). . . . . 1432
Step by step execution mode in textual languages (Structured Text or Instruction List). . . . . 1434
Insertion of a Watchpoint in Textual Languages (Structured Text or Instruction List). . . . . 1437
38.5 Debugging in Function Block Diagram (FBD) Language . . . . . 1440
Inserting a breakpoint in Function Block Diagram language . . . . . 1441
Step by step execution mode in Function Block Diagram language (FBD) . . . . . 1444
Inserting a watchpoint in Function Block Diagram language (FBD). . . 1447
38.6 Debugging User Function Block (DFB) Instances . . . . . 1450
Inserting a breakpoint in a DFB instance (user function block) . . . . . 1451
Step by step execution mode in a DFB instance (user function block) 1453
Inserting a watchpoint in a user function block (DFB) . . . . . 1454
38.7 Debugging in Sequential Function Chart Language (SFC) . . . . . 1455
Inserting a breakpoint in Sequential Function Chart language (SFC) . 1456
Step by step execution mode in Sequential Function Chart language (SFC) . . . . . 1459
Inserting a watchpoint in Sequential Function Chart language . . . . . 1463
38.8 Information on Debugging . . . . . 1464
Tracing task execution . . . . . 1465
Task states during debugging . . . . . 1467
Disconnection during the debug phase . . . . . 1468
Operation of the alarm relay in Debug mode . . . . . 1469

38.9 PLC debug screen . . . 1470
Overview of the CPU debugging screen . . . 1471
Description of the Task tab: Task checking field . . . 1473
Description of the Task tab: Operating mode field . . . 1475
Description of tab: Realtime Clock . . . 1477
Description of the Tab: Information . . . 1478
**Chapter 39 Viewing and adjusting variables . . . 1483**
39.1 Introduction to variable animation . . . 1484
Displaying variables in debug mode . . . 1484
39.2 Variable Display Window . . . 1485
Introduction to the Variable Output Window . . . 1485
39.3 Variable Display Boxes . . . 1488
Introduction to variable display boxes . . . 1488
39.4 Animation Tables . . . 1491
Permanent and Temporary Animation Tables . . . 1492
Introduction to Animation Tables . . . 1494
Managing Animation Tables . . . 1496
Multiple Modifications of Animation Table Variables . . . 1500
Animation Table Properties . . . 1501
Basic functions . . . 1502
Animation of Strings in the Animation Table . . . 1506
Modification Mode . . . 1507
Force Mode . . . 1512
Multiple modification and forcing of Boolean variables . . . 1518
Animation Table of Forced Bits . . . 1521
39.5 Instruction Editor (LL984) . . . 1522
Instruction Editor . . . 1522
39.6 Modifying values: . . . 1525
Modifying values: . . . 1525
**Chapter 40 Machine-Process Simulator . . . 1529**
Introduction . . . 1529
**Part VII Diagnostics . . . 1531**
**Chapter 41 Introduction to diagnostics tools . . . 1533**
Diagnostics with Unity Pro . . . 1533
**Chapter 42 Diagnostics screens . . . 1535**
42.1 Module and processor diagnostics screens . . . 1536
Viewing fault diagnostics . . . 1537
Lists of fault diagnostics . . . 1539

**Chapter 43 Diagnostics Viewer . . . . . 1541**
Reminder of the Diagnostics Associated with the Viewer . . . . . 1542
Operation of the diagnostics in Unity Pro . . . . . 1544
Description of the Diagnostics Viewer . . . . . 1545
How to find causes . . . . . 1548
Configuring the display . . . . . 1550
Managing error messages . . . . . 1552
Diagnostics Viewer Commands . . . . . 1554
DiagViewer Properties . . . . . 1556
Archiving error messages . . . . . 1558
**Chapter 44 System diagnostics . . . . . 1561**
System diagnostic . . . . . 1561
**Chapter 45 Project diagnostics . . . . . 1565**
How to Create a User Diagnostic DFB . . . . . 1566
Project diagnostic . . . . . 1568
**Part VIII Operator screen . . . . . 1571**
**Chapter 46 General information . . . . . 1573**
Introduction . . . . . 1574
Creation of Runtime Screens . . . . . 1575
**Chapter 47 First steps . . . . . 1577**
47.1 Working principles . . . . . 1578
How to access the operator screens . . . . . 1578
47.2 Menus and tools in the runtime screen editor . . . . . 1580
Menus in the operator screen editor . . . . . 1581
Edit toolbar . . . . . 1584
47.3 Configuring options from the runtime screen editor . . . . . 1587
General configuration of operator screens . . . . . 1587
**Chapter 48 Editing runtime screens . . . . . 1589**
48.1 Editing screens from the browser . . . . . 1590
Editing operator screens . . . . . 1591
Operator screen editor functions . . . . . 1593
Screen Family Editor Functions . . . . . 1596
Operator screen properties . . . . . 1598
Properties of Operator Screen Families . . . . . 1601

48.2 Creating an object . . . . . 1602
Introduction of a graphic screen object . . . . . 1603
How to create standard objects . . . . . 1604
How to insert an image in a screen . . . . . 1606
How to create driving objects . . . . . 1608
Operation of the driving objects . . . . . 1610
48.3 Default Properties screen . . . . . 1614
Default Properties screen . . . . . 1614
48.4 Modification of an object's attributes . . . . . 1616
How to access an object's attributes . . . . . 1617
Design tab . . . . . 1618
Extended colors . . . . . 1620
Text tab . . . . . 1621
Animation Tab . . . . . 1623
Animation type tab . . . . . 1625
Trend diagrams . . . . . 1628
Bar chart . . . . . 1630
Screen browse tab . . . . . 1632
Control tab . . . . . 1634
Image Tab . . . . . 1636
Explicit Exchanges Tab . . . . . 1638
Information on the objects . . . . . 1640
The data selection tool . . . . . 1642
48.5 Manipulation of objects in runtime screens . . . . . 1645
How to select graphic objects . . . . . 1646
How to move and resize objects . . . . . 1648
How to delete and copy objects . . . . . 1650
How to position objects . . . . . 1652
How to create compound objects . . . . . 1654
48.6 Managing the runtime screens messages editor . . . . . 1655
Operator screens messages editor . . . . . 1656
How to manage messages linked to operator screens . . . . . 1658
Find dialog box . . . . . 1660
Replace dialog box . . . . . 1662
Message Renumbering Dialog Box . . . . . 1663
Message editing and import/export function . . . . . 1664

**Chapter 49 Runtime screens in online mode . . . . . 1667**
Basic functions in online mode . . . . . 1668
How to select an animated object . . . . . 1669
How to use command objects to modify the variables associated with objects . . . . . 1670
How to view the variables associated with graphic objects . . . . . 1671
How to modify the variables associated with objects from an animation table . . . . . 1672
How to search for cross-references . . . . . 1673
How to manage the display of operator screens . . . . . 1674
How to modify screen control . . . . . 1676
How to obtain information on objects when in connected mode . . . . . 1677
**Chapter 50 Runtime screens objects library . . . . . 1679**
50.1 Managing library objects . . . . . 1680
The Objects Library . . . . . 1681
How to manage the list of graphic items in the library . . . . . 1682
Properties of the object library and its elements . . . . . 1685
Handling the Objects . . . . . 1689
Dereferencing in Operator Screens . . . . . 1691
**Part IX Documentation, Import/export . . . . . 1693**
**Chapter 51 Documentation . . . . . 1695**
Creating Project Documentation . . . . . 1696
Documentation Structure . . . . . 1697
Topics in Documentation . . . . . 1700
Creating Project Documentation . . . . . 1707
Print Preview . . . . . 1709
Page Setup . . . . . 1711
Print . . . . . 1713
Printout Setup . . . . . 1715
**Chapter 52 Import / Export . . . . . 1719**
52.1 General information on the Import/Export function . . . . . 1720
General information on import/export . . . . . 1721
Import / Export Files . . . . . 1722
Type of dialog box for the export function . . . . . 1723
Type of dialog boxes for the import function . . . . . 1726
Managing Conflicts . . . . . 1728

52.2 Importing/Exporting the various elements of Unity Pro . . . . . 1730
Export of the input/output configuration . . . . . 1731
Import of the input/output configuration . . . . . 1732
Exporting the program . . . . . 1733
Program Import . . . . . 1735
Exporting DFB types . . . . . 1737
Importing DFB types . . . . . 1738
Exporting DDTs . . . . . 1739
Importing DDTs . . . . . 1740
Exporting Variables . . . . . 1741
Importing Variables . . . . . 1744
Exporting the communication network configuration . . . . . 1745
Importing the configuration of a communication network . . . . . 1746
Exporting operator screens . . . . . 1747
Importing operator screens . . . . . 1749
Exporting animation tables . . . . . 1750
Importing animation tables . . . . . 1751
Exporting a functional module . . . . . 1752
Importing a functional module . . . . . 1754
Exporting a global project . . . . . 1755
Importing a global project . . . . . 1757
Exporting/Importing LL984 Segments/Networks . . . . . 1759
52.3 Wizard for importing the various elements of Unity Pro . . . . . 1761
Using the assistant . . . . . 1762
Variables tab of the wizard . . . . . 1766
DFB and DDT Tabs of the Wizard . . . . . 1768
SR, Sections, Events and Macro steps tabs of the wizard . . . . . 1770
Wizard animation tables tab . . . . . 1771
Wizard functional module tabs . . . . . 1772
Correspondence file . . . . . 1774
**Part X Conversion Wizard . . . . . 1775**
**Chapter 53 Conversion Wizard . . . . . 1777**
53.1 General Description of the Conversion Wizard . . . . . 1778
Introduction . . . . . 1779
Access to Conversion Wizard . . . . . 1781
General Representation . . . . . 1784
Conversion and Result . . . . . 1786

53.2 Structure Tab of the Conversion Wizard . . . . . 1787
Introduction . . . . . 1788
Selecting Application Parts to be Converted . . . . . 1790
53.3 I/O Remapping Tab of the Conversion Wizard . . . . . 1792
Introduction . . . . . 1793
Remapping Table . . . . . 1796
Remapping Channels, Groups of Channels or Specific I/O Types . . . 1799
Clustering and Remapping of Memory Variables . . . . . 1801
Memory Area Tabs of the Conversion Wizard . . . . . 1803
Concurrent Editing . . . . . 1805
Exclude, Communication -1, Type Restriction . . . . . 1808
53.4 CPU Configuration Tab of the Conversion Wizard . . . . . 1813
Introduction . . . . . 1813
**Appendices . . . . . 1815**
**Appendix A Hyperlinks in Unity Pro . . . . . 1817**
A.1 General . . . . . 1818
Introduction to Hyperlinks . . . . . 1819
Representation and opening of a hyperlink . . . . . 1822
A.2 Principles . . . . . 1823
Definitions . . . . . 1824
Overview of operate-modes . . . . . 1825
1-step mode . . . . . 1826
2-step mode . . . . . 1827
Description of the Hyperlink Comment Window . . . . . 1829
Properties . . . . . 1832
List of barred characters . . . . . 1833
A.3 Hyperlinks in the project browser . . . . . 1834
Creating a hyperlink in the project browser . . . . . 1835
Editing the properties of a hyperlink in the project browser . . . . . 1837
Opening a hyperlink from the project browser . . . . . 1839
Deleting a hyperlink in the project browser . . . . . 1840
A.4 Hyperlink in the data editor . . . . . 1841
Creating a hyperlink in the data editor . . . . . 1842
Editing a hyperlink in the data editor . . . . . 1844
Selecting a hyperlink in the data editor . . . . . 1846
Deleting a hyperlink in the data editor . . . . . 1847

A.5 Hyperlinks in the DFB type attributes . . . . . 1849
Creating a Hyperlink from DFB Type Attributes . . . . . 1850
Modifying a Hyperlink from DFB Type Attributes . . . . . 1853
Selecting a Hyperlink from DFB Type Attributes . . . . . 1855
Deleting a Hyperlink from DFB Type Attributes . . . . . 1857
A.6 Hyperlinks in animation tables . . . . . 1860
Creating a hyperlink in an animation table . . . . . 1861
Editing a hyperlink from an animation table . . . . . 1863
Selecting a hyperlink from an animation table . . . . . 1865
Deleting a hyperlink from an animation table . . . . . 1866
A.7 Hyperlinks in runtime screens . . . . . 1867
Creating a hyperlink from operator screens . . . . . 1868
Editing a hyperlink in the operator screens . . . . . 1870
Selecting a hyperlink in an operator screen . . . . . 1872
Deleting a hyperlink in an operator screen . . . . . 1873
A.8 Hyperlinks in the documentation folder . . . . . 1874
Creating a hyperlink in the general information of the project folder . 1875
Editing a hyperlink in the documentation folder . . . . . 1877
Deleting a hyperlink in the documentation folder . . . . . 1879
A.9 Hyperlinks and drag-and-drop . . . . . 1881
Drag/drop . . . . . 1881
**Appendix B User Interface . . . . . 1885**
Window . . . . . 1886
Anchored window . . . . . 1889
Menus . . . . . 1892
Menu commands . . . . . 1895
Toolbar . . . . . 1898
Output windows . . . . . 1903
Status bar . . . . . 1905
Directory Structures . . . . . 1907
Using the Mouse . . . . . 1909
Using the Keyboard . . . . . 1910
Data selection boxes . . . . . 1911
Dialog boxes . . . . . 1913
**Appendix C Keyboard Shortcuts . . . . . 1917**
C.1 General Keyboard Shortcuts . . . . . 1918
Bus Editor Keyboard Shortcuts . . . . . 1919
Unity Pro Main Screen and Project Browser Keyboard Shortcuts . . . . 1920

C.2 Configuring the Project . . . . . 1922
PLC Configuration Screen Keyboard Shortcuts . . . . . 1922
C.3 Creating the Project . . . . . 1923
ST/IL Editors Keyboard Shortcuts . . . . . 1924
FBD Editor Keyboard Shortcuts . . . . . 1926
SFC Editor Keyboard Shortcuts . . . . . 1928
LD Editor Keyboard Shortcuts . . . . . 1930
Data Editor Keyboard Shortcuts . . . . . 1932
Type Library Browser Keyboard Shortcuts . . . . . 1933
Type Library Manager Keyboard Shortcuts . . . . . 1934
C.4 Operating the Project . . . . . 1935
Search Replace Tool Screen Keyboard Shortcuts . . . . . 1936
Diagnostic Viewer Screen Keyboard Shortcuts . . . . . 1937
PLC Screen Keyboard Shortcuts . . . . . 1938
Animation Screen Keyboard Shortcuts . . . . . 1939
Operator Screen Keyboard Shortcuts . . . . . 1940
C.5 Documenting the Project . . . . . 1941
Documentation Screen Keyboard Shortcuts . . . . . 1941
**Glossary** . . . . . 1943
**Index** . . . . . 1965

# Safety Information

## Important Information

### NOTICE

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



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



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

**⚠ DANGER**

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

**⚠ WARNING**

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

**⚠ CAUTION**

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

***NOTICE***

***NOTICE*** is used to address practices not related to physical injury.

## PLEASE NOTE

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

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

# About the Book

## At a Glance

### Document Scope

This document describes programming a Modicon M340, Modicon M580, Premium, Atrium, or Quantum PLC using the IEC 61131-3 conforming programming languages Function Block Diagram (FBD), Ladder Diagram (LD), Sequential Function Chart (SFC), Instruction List (IL) and Structured Text (ST).

### Validity Note

This document is valid for Unity Pro 11.0 or later.

### Related Documents

| Title of Documentation | Reference Number |
|---|---|
| Unity Pro Program Languages and structure, Reference Manual | 35006144 (English),<br>35006145 (French),<br>35006146 (German),<br>35013361 (Italian),<br>35006147 (Spanish),<br>35013362 (Chinese) |
| Unity Pro System Bits and Words, Reference Manual | EIO0000002135 (English),<br>EIO0000002136 (French),<br>EIO0000002137 (German),<br>EIO0000002138 (Italian),<br>EIO0000002139 (Spanish),<br>EIO0000002140 (Chinese) |

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

# Part I
# General

## Subject of this Part

This part presents Unity Pro services and describes the different tools used to manage a project: access rights, save/restore, online/offline mode services, etc.

## What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 1 | General Presentation of Unity Pro | 35 |
| 2 | Programming Language Editors | 43 |
| 3 | Connecting a terminal to a PLC | 55 |
| 4 | Global project management | 57 |
| 5 | Project browser | 161 |
| 6 | Functional modules | 221 |
| 7 | FDT Container | 253 |
| 8 | Type Library Manager | 293 |

# Chapter 1
# General Presentation of Unity Pro

## Overview

This chapter describes the general features of a project created using Unity Pro.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Capabilities of Unity Pro | 36 |
| User Interface | 40 |

# Capabilities of Unity Pro

## Hardware Platforms

Unity Pro supports the following hardware platforms:

- Modicon M340
- Modicon M580
- Quantum
- Momentum
- Premium
- Atrium

## Programming Languages

Unity Pro provides the following programming languages for creating the user program:

- Function Block Diagram FBD
- Ladder Diagram (LD) language
- Instruction List IL
- Structured Text ST
- Sequential Control SFC
- Ladder Logic 984 (LL984)

All of these programming languages can be used together in the same project.

All these languages (except LL984) conform to IEC 61131-3.

## Block Libraries

The blocks that are included in the delivery of Unity Pro extensive block libraries extend from blocks for simple boolean operations, through blocks for strings and array operations to blocks for controlling complex control loops.

For a better overview, the different blocks are arranged in libraries, which are then broken down into families.

The blocks can be used in the programming languages FBD, LD, IL, and ST.

## Elements of a Program

A program can be constructed from:

- a master task (MAST)
- a FAST task (not available for Momentum)
- one to 4 AUX tasks (not available for Modicon M340 and Momentum)
- sections, which are assigned one of the defined tasks
- sections for processing time controlled events (Timerx, not available for Momentum)
- sections for processing hardware controlled events (EVTx, not available for Momentum)
- subroutine sections (SR)

## Software Packages

The following software packages are available:

- Unity Pro S
- Unity Pro L
- Unity Pro XL
- Unity Pro XLS

## Performance Scope

This table shows the main characteristics of the individual software packages:

| | Unity Pro S | Unity Pro L | Unity Pro XL | Unity Pro XLS |
|---|---|---|---|---|
| **Programming languages** | | | | |
| Function Block Diagram FBD | + | + | + | + |
| Ladder Diagram (LD) language | + | + | + | + |
| Instruction List IL | + | + | + | +(2) |
| Structured Text ST | + | + | + | +(2) |
| Sequential Language SFC | + | + | + | +(2) |
| Ladder Logic 984 (LL984) | + | + | + | + |
| **Libraries (1)** | | | | |
| Standard library | + | + | + | +(2) |
| Control library | + | + | + | +(2) |
| Communication library | + | + | + | +(2) |
| Diagnostics library | + | + | + | +(2) |
| I/O management library | + | + | + | +(2) |
| System library | + | + | + | +(2) |
| Motion control drive library | - | + | + | +(2) |
| TCP Open library | - | optional | optional | optional (2) |
| Obsolete library | + | + | + | +(2) |
| MFB library | + | + | + | +(2) |
| Safety library | - | - | - | + |
| Memory card file management library | + | + | + | +(2) |
| **General information** | | | | |
| Create and use data structures (DDTs) | + | + | + | +(2) |
| Create and use Derived Function Blocks (DFBs) | + | + | + | + |

| | Unity Pro S | Unity Pro L | Unity Pro XL | Unity Pro XLS |
|---|---|---|---|---|
| Project browser with structural and/or functional view | + | + | + | + |
| Managing access rights | + | + | + | + |
| Operator screen | + | + | + | + |
| Diagnostic viewer | + | + | + | + |
| System diagnostics | + | + | + | + |
| Project diagnostics | + | + | + | +(2) |
| Trending Tool | + | + | + | + |
| Application converter | PL7 converter<br>Concept Converter<br>Partial conversion | PL7 converter<br>Concept Converter | PL7 converter<br>Concept Converter | PL7 converter<br>Concept Converter |
| Managing multi-stations | - | - | - | - |
| **Supported platforms** | | | | |
| Modicon M340 | BMX P34 1000<br>BMX P34 20•• | BMX P34 1000<br>BMX P34 20•• | BMX P34 1000<br>BMX P34 20•• | BMX P34 1000<br>BMX P34 20•• |
| Modicon M580 | - | BME P58 10••<br>BME P58 20••<br>BME P58 30••<br>BME H58 20•• | BME P58 10••<br>BME P58 20••<br>BME P58 30••<br>BME H58 20••<br>BME P58 40••<br>BME H58 40••<br>BME P58 50••<br>BME P58 60••<br>BME H58 60•• | - |
| Momentum | 171CBU78090<br>171CBU98090<br>171CBU98091 | 171CBU78090<br>171CBU98090<br>171CBU98091 | 171CBU78090<br>171CBU98090<br>171CBU98091 | 171CBU78090<br>171CBU98090<br>171CBU98091 |
| Premium | - | All CPUs **except**:<br>P57 554M<br>P57 5634M<br>P57 6634M | All CPUs | All CPUs |

| | Unity Pro S | Unity Pro L | Unity Pro XL | Unity Pro XLS |
|---|---|---|---|---|
| Quantum | - | 140 CPU 311 10<br>140 CPU 434 12 U/A*<br>140 CPU 534 14 U/A*<br>* Upgrade using Unity OS | CPU 311 10<br>CPU 434 12 U/A<br>CPU 534 14 U/A<br>CPU 651 50<br>CPU 651 60<br>CPU 652 60<br>CPU 658 60<br>CPU 670 60<br>CPU 671 60<br>CPU 672 60<br>CPU 672 61<br>CPU 678 61 | CPU 311 10<br>CPU 434 12 U/A<br>CPU 534 14 U/A<br>CPU 651 50<br>CPU 651 60<br>CPU 652 60<br>CPU 658 60<br>CPU 670 60<br>CPU 671 60<br>CPU 672 60<br>CPU 672 61<br>CPU 678 61<br>CPU 651 60S<br>CPU 671 60S |
| Atrium | - | All CPUs | All CPUs | All CPUs |
| Simulator | + | + | + | + |
| **Openness** | | | | |
| Hyperlinks | + | + | + | + |
| Unity Pro Server (for OFS, UAG) | + | + | + | + |
| **Software components contained in the software package** | | | | |
| Documentation as context help and PDF | + | + | + | + |
| OS Loader tool + HW firmware | + | + | + | + |
| Unity Loader | + | + | + | + |

Legend:
**+** = available
**+ (1)** = Availability of the blocks depends on the hardware platforms *(see Unity Pro, Standard, Block Library)*.
**+ (2)** = Available on all PLC except platforms CPU 651 60S, CPU 671 60S.
**-** = not available

## Naming Convention

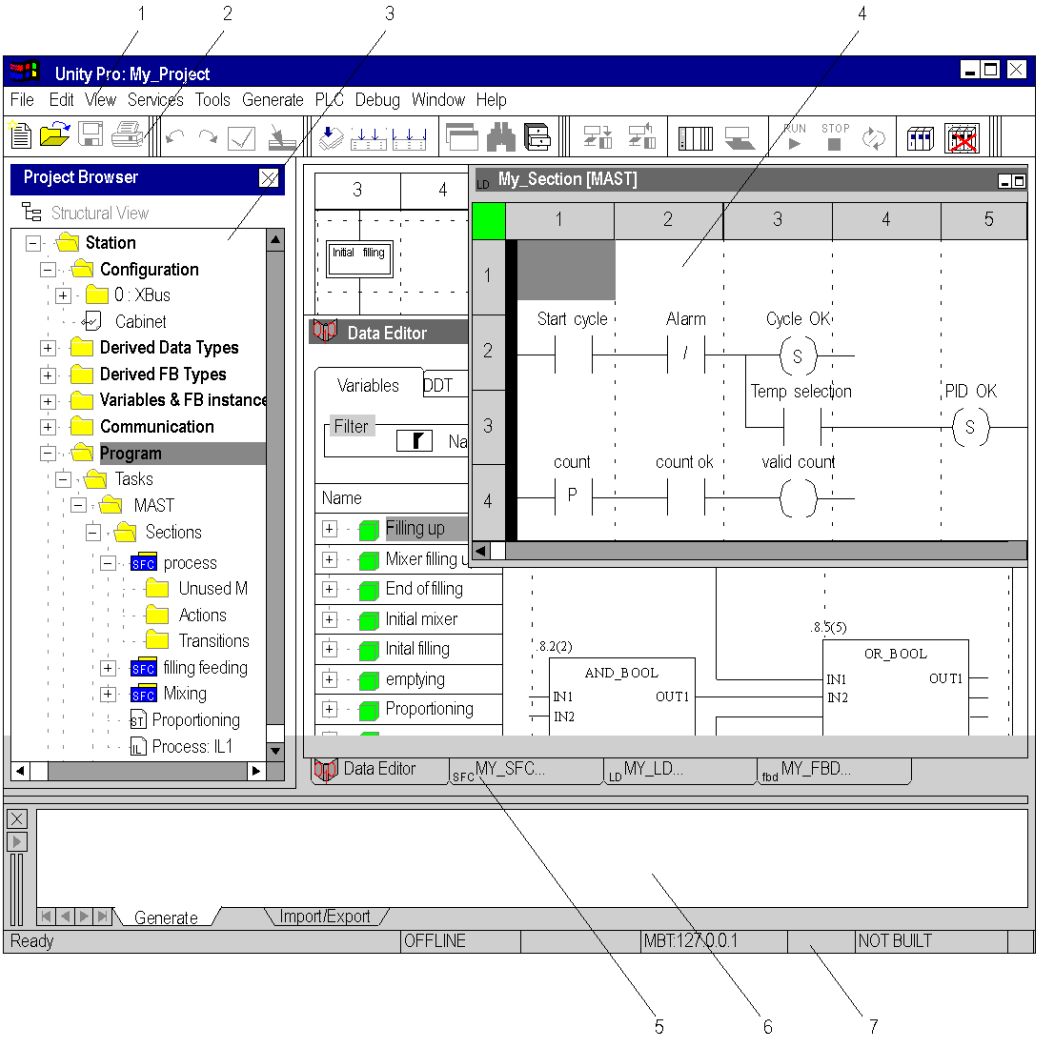
In the rest of this document, "Unity Pro" is used as general term for "Unity Pro S", "Unity Pro L", "Unity Pro XL" and "Unity Pro XLS".

# User Interface

## Overview

The user interface consists of several, configurable windows and toolbars.

User interface:



Legend:

| Number | Description |
|---|---|
| 1 | Menu bar *(see page 1892)* |
| 2 | Toolbar *(see page 1898)* |
| 3 | Project Browser *(see page 161)* |
| 4 | Editor window (programming language editors, data editor, etc.) |
| 5 | Register tabs for direct access to the editor window |
| 6 | Information window *(see page 1903)* (provides information about errors which have occurred, signal tracking, import functions, etc.) |
| 7 | Status bar *(see page 1905)* |

# Chapter 2
# Programming Language Editors

## Overview

This chapter describes the different programming languages provided by Unity Pro.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Function Block Diagram FBD | 44 |
| Ladder Diagram (LD) Language | 46 |
| General Information about SFC Sequence Language | 48 |
| Instruction List IL | 52 |
| Structured Text ST | 53 |

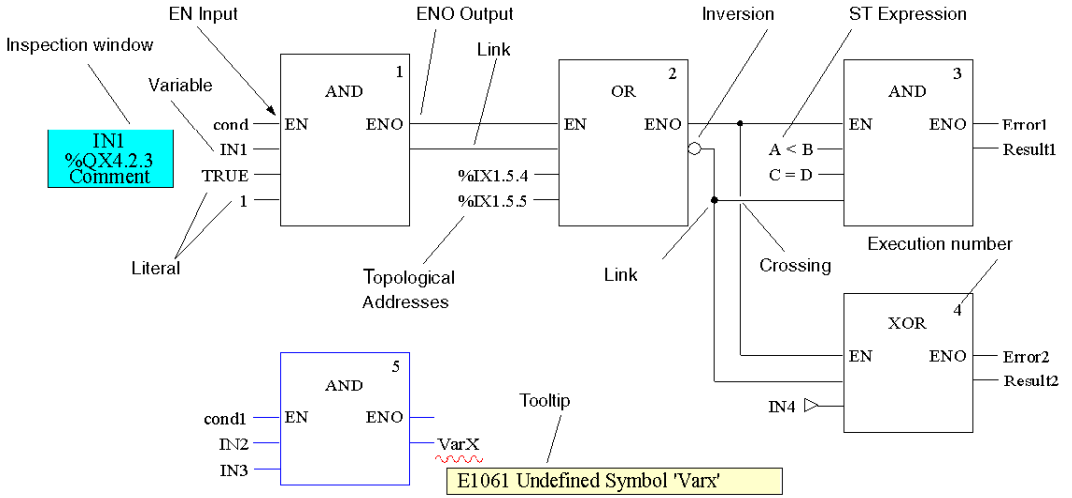
# Function Block Diagram FBD

## Introduction

The FBD editor is used for graphical function block programming according to IEC 61131-3.

## Representation

Representation of an FBD section:




## Objects

The objects of the FBD (Function Block Diagram) programming language help to divide a section into a number of:

- Elementary Functions (EFs),
- Elementary Function Blocks (EFBs)
- Derived Function Blocks (DFBs)
- Procedures
- Subroutine calls
- Jumps
- Links
- Actual Parameters
- Text objects to comment on the logic

## Properties

FBD sections have a grid behind them. A grid unit consists of 10 coordinates. A grid unit is the smallest possible space between 2 objects in an FBD section.

The FBD programming language is not cell oriented but the objects are still aligned with the grid coordinates.

An FBD section can be configured in number of cells (horizontal grid coordinates and vertical grid coordinates).

The program can be entered using the mouse or the keyboard.

## Input Aids

The FBD editor offers the following input aids:

- Toolbars for quick and easy access to the desired objects
- Syntax and semantics are checked as the program is being written.
  - Incorrect functions and function blocks are displayed in blue
  - Unknown words (e.g. undeclared variables) or unsuitable data types are marked with a red wavy line
  - Brief description of errors in the Quickinfo (Tooltip)
- Information for variables and pins can be displayed in a Quickinfo (Tooltip)
  - type, name, address and comment of a variable/expression
  - type, name and comment of an FFB pin
- Tabular display of FFBs
- Actual parameters can be entered and displayed as symbols or topological addresses
- Different zoom factors
- Tracking of links
- Optimization of link routes
- Display of inspection windows

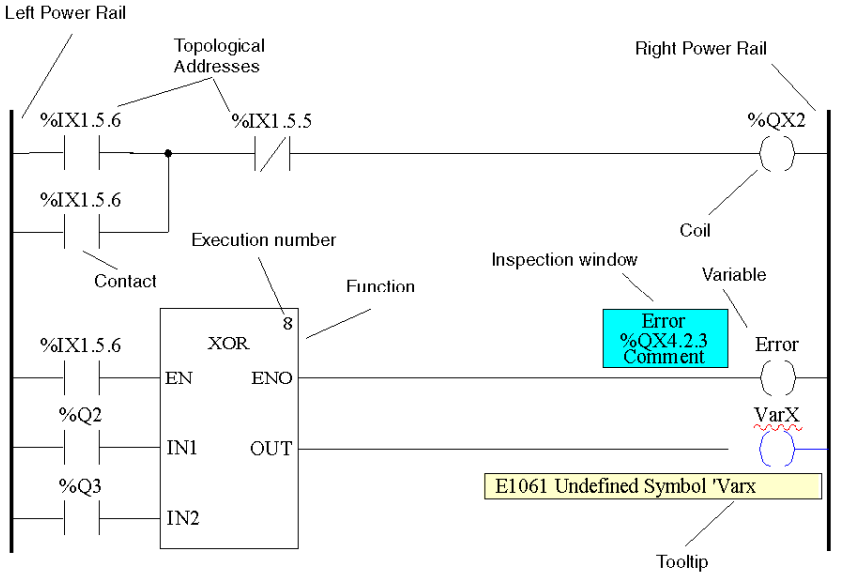
# Ladder Diagram (LD) Language

## Introduction

The LD editor is used for graphical ladder diagram programming according to IEC 61131-3.

## Representation

Representation of an LD section:




## Objects

The objects of the LD programming language help to divide a section into a number of:

- Contacts,
- Coils,
- Elementary Functions (EFs)
- Elementary Function Blocks (EFBs),
- Derived Function Blocks (DFBs)
- Procedures
- Control elements
- Operation and compare blocks which represent an extension to IEC 61131-3
- Subroutine calls
- Jumps
- Links
- Actual Parameters
- Text objects to comment on the logic

## Properties

LD sections have a background grid that divides the section into lines and columns.

The LD programming language is cell oriented, i.e. only one object can be placed in each cell.

LD sections can be 11-64 columns and 17-2000 lines in size.

The program can be entered using the mouse or the keyboard.

## Input Aids

The LD editor offers the following input aids:

- Objects can be selected from the toolbar, the menu or directly using shortcut keys
- Syntax and semantics are checked as the program is being written.
  - Incorrect objects are displayed in blue
  - Unknown words (e.g. undeclared variables) or unsuitable data types are marked with a red wavy line
  - Brief description of errors in the Quickinfo (Tooltip)
- Information for variables and for elements of an LD section, that can be connected to a variable (pins, contacts, coils, operation and compare blocks), can be displayed in a Quickinfo (Tooltip)
  - type, name, address and comment of a variable/expression
  - type, name and comment of FFB pins, contacts etc.
- Tabular display of FFBs
- Actual parameters can be entered and displayed as symbols or topological addresses
- Different zoom factors
- Tracking of FFB links
- Optimizing the link routes of FFB links
- Display of inspection windows

# General Information about SFC Sequence Language

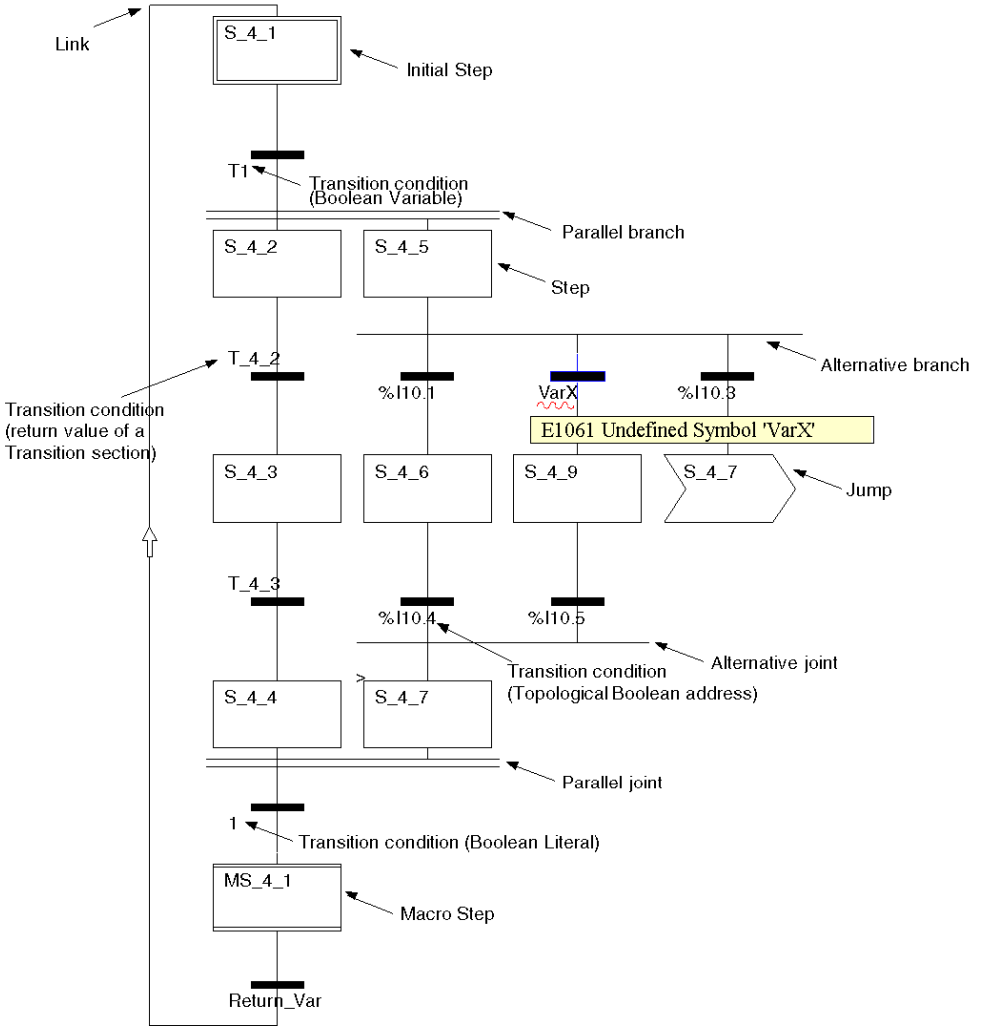
## Introduction

The sequence language SFC (Sequential Function Chart), which conforms to IEC 61131-3, is described in this section.

IEC conformity restrictions can be lifted through explicit enable procedures. Features such as multi token, multiple initial steps, jumps to and from parallel strings etc. are then possible.

## Representation

Representation of an SFC section:



### Objects

An SFC section provides the following objects for creating a program:

- Steps
- Macro steps (embedded sub-step sequences)
- Transitions (transition conditions)
- Transition sections
- Action sections
- Jumps
- Links
- Alternative sequences
- Parallel sequences
- Text objects to comment on the logic

### Properties

The SFC editor has a background grid that divides the section into 200 rows and 32 columns.

The program can be entered using the mouse or the keyboard.

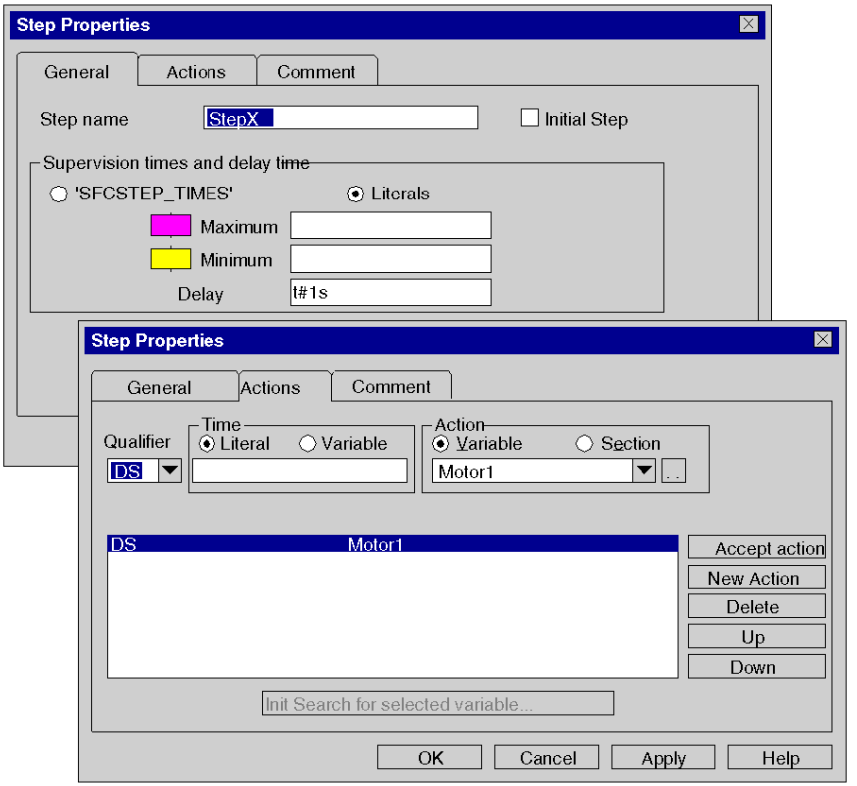
### Input Aids

The SFC editor offers the following input aids:

- Toolbars for quick and easy access to the desired objects
- Automatic step numbering
- Direct access to actions and transition conditions
- Syntax and semantics are checked as the program is being written.
  - Incorrect objects are displayed in blue
  - Unknown words (e.g. undeclared variables) or unsuitable data types are marked with a red wavy line
  - Brief description of errors in the Quickinfo (Tooltip)
- Information for variables and for transitions can be displayed in a Quickinfo (Tooltip)
  - type, name, address and comment of a variable/expression
  - type, name and comment of transitions
- Different zoom factors
- Show/hide the allocated actions
- Tracking of links
- Optimization of link routes

## Step Properties

Step properties:




The step properties are defined using a dialog box that offers the following features:

- Definition of initial steps
- Definition of diagnostics times
- Step comments
- Allocation of actions and their qualifiers

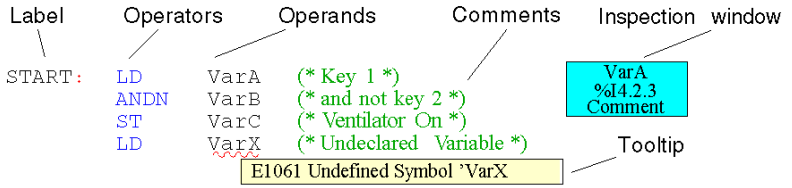
# Instruction List IL

## Introduction

The IL editor is used for instruction list programming according to IEC 61131-3.

## Representation

Representation of an IL section:




## Objects

An instruction list is composed of a series of instructions.

Each instruction begins on a new line and consists of:

- An operator
- A modifier if required
- One or more operands if required
- A label as a jump target if required
- A comment about the logic if required.

## Input Aids

The IL editor offers the following input aids:

- Syntax and semantics are checked as the program is being written.
  - Keywords and comments are displayed in color
  - Unknown words (e.g. undeclared variables) or unsuitable data types are marked with a red wavy line
  - Brief description of errors in the Quickinfo (Tooltip)
- Tabular display of the functions and function blocks
- Input assistance for functions and function blocks
- Operands can be entered and displayed as symbols or topological addresses
- Display of inspection windows

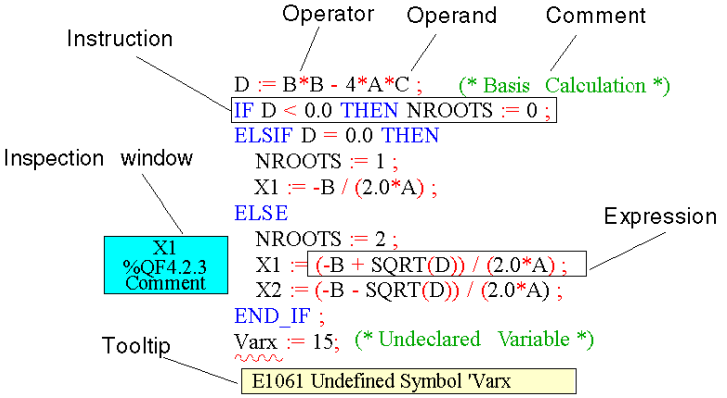
# Structured Text ST

## Introduction

The ST editor is used for programming in structured text according to IEC 61131-3.

## Representation

Representation of an ST section:




## Objects

The ST programming language works with "Expressions".

Expressions are constructions consisting of operators and operands that return a value when executed.

Operators are symbols representing the operations to be executed.

Operators are used for operands. Operands are variables, literals, function and function block inputs/outputs etc.

Instructions are used to structure and control the expressions.

## Input Aids

The ST editor offers the following input aids:

- Syntax and semantics are checked as the program is being written.
  - Keywords and comments are displayed in color
  - Unknown words (e.g. undeclared variables) or unsuitable data types are marked with a red wavy line
  - Brief description of errors in the Quickinfo (Tooltip)
- Tabular display of the functions and function blocks
- Input assistance for functions and function blocks
- Operands can be entered and displayed as symbols or topological addresses
- Display of inspection windows

# Chapter 3
# Connecting a terminal to a PLC

## Methodology for connecting from a PC to a PLC

### At a Glance

To program a Modicon M340, Modicon M580, Premium, or Quantum PLC using Unity Pro software, observe the rules described below.

### Cables References

Select a cable that conforms to your PLC type to physically connect your PLC to your computer:

| Platform | Produce | Product Reference |
|---|---|---|
| Quantum | Modbus cable | 990 NAA 263 20 |
| | Modbus Plus cable | UNY XCA 42 020 |
| | crossed Ethernet cable | 490 NTC 000 05(1) |
| | USB cable | UNY XCA USB 033(2) |
| Premium | terminal port cable | TSX PCX 1031 (RS 232/RS 485) |
| | terminal port cable | TSX PCX 3030 (USB/RS 485) |
| | crossed Ethernet cable | 490 NTC 000 05(1) |
| | USB cable | UNY XCA USB 033(2) |
| Modicon M340 | Modbus cable | TCS MCN 3M4F3C2 |
| | crossed Ethernet cable | 490 NTC 000 05(1) |
| | USB cable | BMX XCA USBH •••(2) |

**(1)** Use a crossed cable for a PC-PLC link; whereas, use a straight-through cable for PC-Hub or PLC-Hub link.

**(2)** When connecting a PC to a PLC, the use of a USB 2.0 shielded cable following the USB international standard is strongly recommended. The cables mentioned earlier avoid unexpected behavior of the PLC as they are shielded and tested against electrical noises.

**(3)** This cable is usable but not recommended as all ports of a M580 system are Auto-MDIX capable and crossed cables could be incompatible with future gigabit systems.

| Platform | Produce | Product Reference |
|---|---|---|
| Modicon M580 | mini-USB to USB-A programming cable (industrial use rated) | BMX XCA USBH •••(2) |
| | Ethernet straight-through Cat5E cable, CE or UL standard (industrial use rated) | 490 NTW 000 ••(U) |
| | Ethernet straight-through Cat5E cable, CE or UL standard (industrial use/high demand rating) | TCSEC(E/U)3M3M••S4 |
| | Ethernet straight-through Cat5E cable, 1 meter | TCSECN3M3M1S4 |
| | Ethernet crossed Cat5E cable (industrial use rated), CE or UL standard. (3) | 490 NTC 000 ••(U) |

**(1)** Use a crossed cable for a PC-PLC link; whereas, use a straight-through cable for PC-Hub or PLC-Hub link.
**(2)** When connecting a PC to a PLC, the use of a USB 2.0 shielded cable following the USB international standard is strongly recommended. The cables mentioned earlier avoid unexpected behavior of the PLC as they are shielded and tested against electrical noises.
**(3)** This cable is usable but not recommended as all ports of a M580 system are Auto-MDIX capable and crossed cables could be incompatible with future gigabit systems.

## Procedure

The table below describes how to connect from a PC to a PLC.

| Step | Action |
|---|---|
| 1 | Physically connect your PLC to your computer using the appropriate cables *(see page 55)*. |
| 2 | Check that for the chosen communication protocol, the Driver corresponding to your operating system is correctly installed.<br>**Note**: The documentation on drivers *(see Communication Drivers, Installation Manual)* describes the various drivers available according to the operating system. For example, for the PC to be able to use an X-Way address on an Ethernet network, you must install the XIP driver. |
| 3 | Verify in the toolbar that the simulator icon is unchecked. |
| 4 | Choose the address of the PLC *(see page 101)* to which you want to connect and the type of link *(see page 107)* using the **PLC → Set Address...** command. |
| 5 | Connect by selecting the command: **PLC → Connection**<br>**Result**: You can now perform all the operations available on line, set the PLC to RUN or to STOP, download projects, or make program modifications, etc. |

# Chapter 4
# Global project management

## Subject of this Chapter

This chapter introduces the various tools used for project management: security and user access rights management, project creation, project backup, transfer of the project to the PLC, management of the user memory, etc.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 4.1 | Access security management | 58 |
| 4.2 | Project management | 80 |

# Section 4.1
## Access security management

### Subject of this Section

This section introduces the tools used to manage security for a project.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Unity Pro access security management | 59 |
| Security editor | 61 |
| User Information | 62 |
| User Functions | 64 |
| User Profiles | 67 |
| Policies | 72 |
| Recovery procedure with an error | 74 |
| Access rights | 75 |

# Unity Pro access security management

## At a Glance

A security configuration tool lets you limit and control access to the software different functionalities.

**NOTE:** Protected access to the software is optional.

Unity Pro access security concerns the terminal on which the software is installed and not the project, which has its own protection system.

A log file can be used to keep a chronological record of the various operations carried out by users with access to the software.

## Super User (Supervisor)

The super user is the only person with rights to manage access security for the software. The super user defines the list including the names of users who can access the software and their access rights.

During installation of the software on the drop, only the super user can access the security configuration without any limitation of rights (without a password).

**NOTE:** The user name reserved for the super user is Supervisor.

The super user has the following rights:

- create or modify the user list,
- create or modify user profiles,
- disable one or more users,
- modify the rules for software access security,
- modify his password,
- reset user passwords.

## Users

Software users are defined in the list of users by the super user, if Unity Pro access security is active. If your name is in the user list, you can access a software instance by entering your name (exactly as it appears on the list) and your password.

A user has the following rights:

- access the rights defined by his user profile in read mode,
- modify his password.

## User Profile

The profile for a user comprises all of his access rights. It is defined by a name (2 to16 characters), an optional comment (maximum of 256 characters) and a list of access rights. Unity Pro provides 5 preconfigured user profiles that cannot be modified. To complete this list, the super user can create all the personalized profiles that he requires.

### Preconfigured User Profiles

Unity Pro provides the following 5 user profiles:

| Profile | Description |
|---|---|
| **ReadOnly** | The user can only access the project in read mode, except for the PLC address, which can be modified. He can also copy or download the project. |
| **Operate** | The user has the same rights as with a **ReadOnly** profile, with the added possibility of modifying execution parameters (constants, initial values, task cycle times, etc.). |
| **Adjust** | The user has the same rights as with an **Operate** profile, with the added possibility of uploading a project (transfer to the PLC) and modifying the PLC operating mode (**Run**, **Stop**, ...). |
| **Debug** | The user has the same rights as with an **Adjust** profile, with the added possibility of using the debugging tools. |
| **Program** | The user has the same rights as with a **Debug** profile, with the added possibility of modifying the program. |
| **Disabled** | User cannot access the project. |

### Default User Profile

When the software access security is active, the super user can authorize a given user to open the software instance without having to enter his name and password. In this case, this user has a default profile.

### Log Information Generated by the Security Editor

Security editor log information is managed by the Microsoft Windows **Event Viewer**. If you want to display the log information, launch the event viewer. Here you can filter according to all the **server sequential** events.

You can use all the Event viewer functions, like **sort**, **filter** etc.

### Confirm

If this option is chosen by the super user, enter your name and password each time secure access is defined for a user action.

### Multiple Instancing

When Unity Pro executes multiple instances concurrently on the same terminal, each of these instances grants access to the functions that were defined by the rights of the user who created this instance. However, the Unity Pro access method used (inactive security or type of active security) is identical for all the instances.

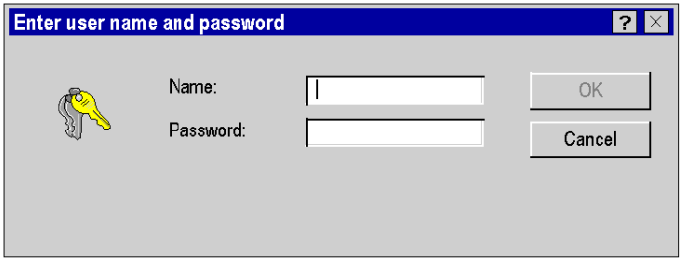
# Security editor

## At a Glance

The security editor is used to define software users and their access rights. It also allows you to define actions which are protected (user name and password) and which are stored in the log file.

## Launching the security editor.

Execute the following commands to start security editor:

| Step | Action |
|---|---|
| 1 | Open with **Start** → **Programs** → **Schneider Electric** → **SoCollaborative** → **Unity Pro** → **Security Editor**.The following dialog appears:<br>  |
| 2 | Enter your name and password. |
| 3 | Confirm with **OK** . |
| 4 | If you are a user *(see page 59)*, you can access the User information screen *(see page 62)* which allows you to consult your user profile or to modify your password. |
| 5 | If you are the Super User (Supervisor) *(see page 59)* the Security Editor *(see page 62)* will launch enabling administration of Unity Pro access security. |

**NOTE:** The user name reserved for the super user is Supervisor.

# User Information

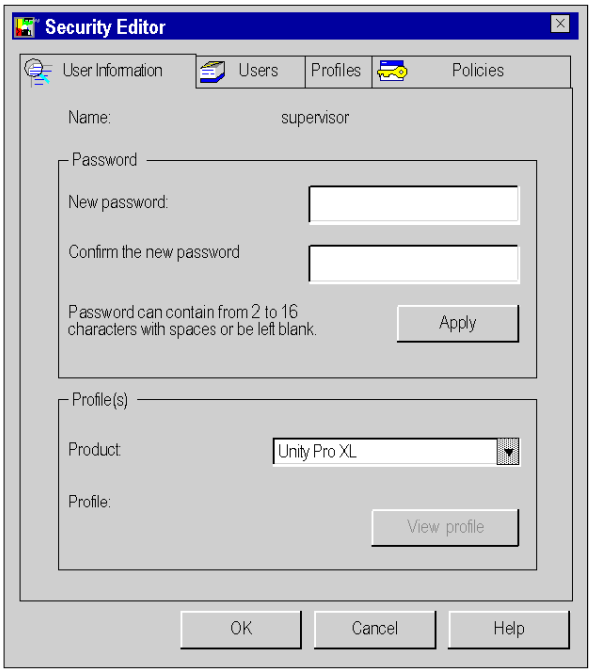
## At a Glance

This tab in the security editor can be accessed by all users. It enables the execution of the following functions:

- modify a password. Users, including the super user, can only modify their own passwords,
- consult a profile and the associated user rights.

## User information tab

The **User Information** looks as follows. It indicates the name of the user and contains the two sections **Password** and **Profile**:



## Description of the parameters

**Password** field,

| Parameter | Description |
|---|---|
| **New password** | Data entry field for the new password: 2 to 16 characters. This field can be empty (no characters entered). |
| **Confirm the new password** | Confirmation field for the new password. The contents of this field has to be identical to that of the **New password** field. |
| **Apply** | Command used to apply the new password.<br>**Note**: The new password is entirely acknowledged only if it is validated with the **OK** command. Clicking **Cancel** cancels acknowledgment of the new password. |

**Profile(s)** field

| Parameter | Description |
|---|---|
| **Product** | Enables the selection of the product for which you want to display your user profile: In our case, Unity Pro. |
| **Profile** | Shows the name of your user profile for the product. |
| **View profile** | Use this command to display all of the access rights included in your user profile. |

# User Functions

## At a Glance

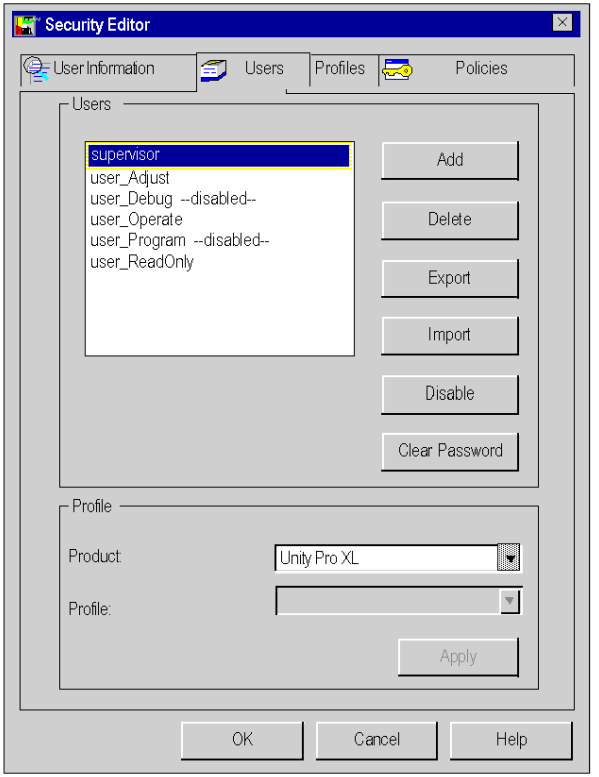
Only the Super User *(see page 59)* can access this tab. It enables the execution of the following functions:

- Modify the user list.
- Assign a profile to each user.
- Import/export information for one or more users.
- Disable/enable one or more users.
- Delete passwords for one or more users.

**NOTE:** The user name reserved for the super user is Supervisor.

## Users tab

The **User** tab looks like this. It contains the two areas **User** and **Profile**:



## Description of the parameters

### **Users** field

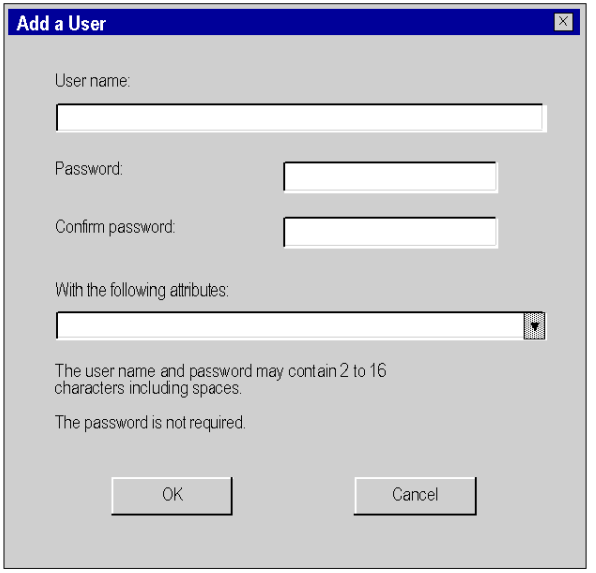
| Parameter | Description |
|---|---|
| **Users** | List of users with access to Unity Pro. The name of the super user is Supervisor.<br>The Preconfigured User *(see page 60)* field is also available for selection:<br>● **user_Adjust**,<br>● **user_Debug**,<br>● **user_Operate**,<br>● **user_Program**,<br>● **user_ReadOnly**. |
| **Add** | Command used to add a new user to the list. |
| **Delete** | Command used to delete the selected user(s) from the list. |
| **Export** | Command used to export the information (name and profiles) of the selected user(s) to a file. Passwords are not exported. |
| **Import** | Command used to import the information (name and profiles) of the selected user(s) from a file. If a user is already on the list, a warning message is displayed. |
| **Disable/Enable** | Command used to disable selected users. A disabled user is blocked during login. One or more users can be selected. You are asked whether the user or users should really be disabled prior to disabling them. **Disable** is not possible for the user Supervisor.<br>If a user has been disabled, the user name is grayed out and **--disabled--** is displayed in the profile.<br>If a disabled user attempts to start Unity Pro, the **User disabled** message appears. |
| **Clear Password** | Command used to delete passwords for selected users such as when a user forgot his password. **Clear Password** is not possible for predefined users. |

### **Profiles** field

| Parameter | Description |
|---|---|
| **Product** | Used to choose the product whose user profile you would like to define. |
| **Profile** | Used to choose the profile that is assigned to the selected user. |
| **Apply** | Command used to apply the profile assignment. The profile is entirely acknowledged only if it is validated with the **OK** command. Clicking **Cancel** cancels assignment of the profile. |

## Adding a user

Clicking **Add** displays the following dialog window:




2 data entry fields are provided:

| Parameter | Description |
|---|---|
| **User name** | This field is used to enter the name of the new user (2 to 16 characters). If the name entered is not correct, or if it exists already, a warning message is displayed. |
| **Password** | Data entry field for the password: 2 to 16 characters. This field can be empty (no characters entered). |
| **Confirm password** | Confirmation field for the password. The contents of this field has to be identical to that of the **Password** field. |
| **With the following attributes** | Field used to select the user whose attributes you would like to retrieve. |

# User Profiles

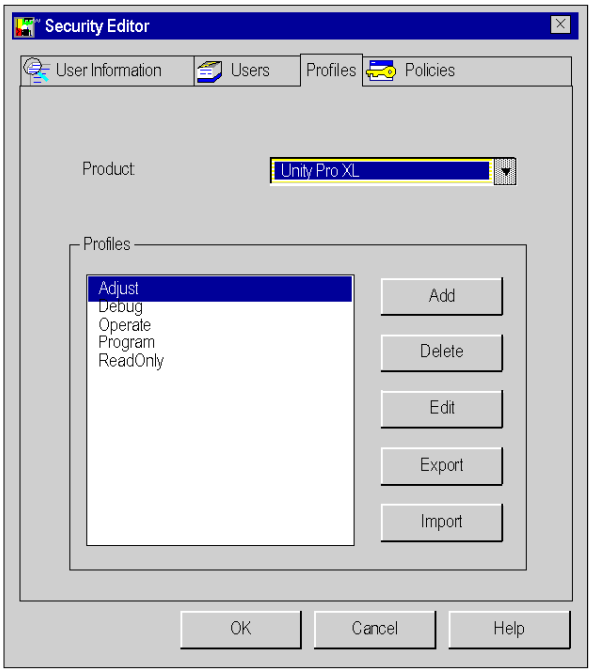
## At a Glance

Only the Super User *(see page 59)* can access this tab. It is used to perform the following functions:

- Add or delete a profile in the list.
- Read and modify the access rights associated with a profile.
- Import/export information for one or more user profiles.

## “Profiles” Tab

The **Profiles** tab looks as follows. It features the **Product** and **Profiles** areas:



## Description of the parameters

**Product** area:

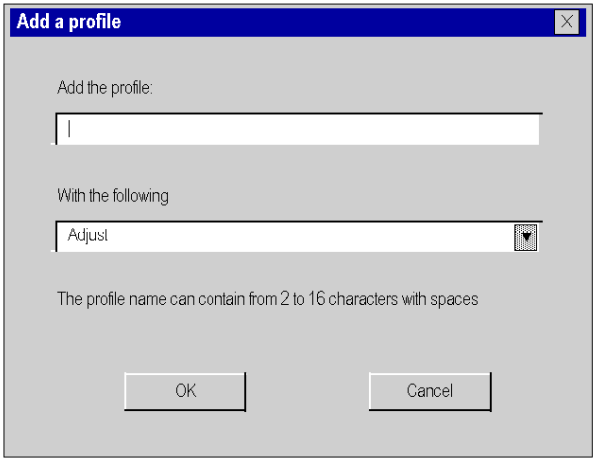
| Parameter | Description |
|---|---|
| **Product** | Used to specify the product whose user profile you would like to access. |

**Profiles** area:

| Parameter | Description |
|---|---|
| **Profiles** | When the product chosen is Unity Pro,list of user profiles associated with Unity Pro. |
| **Add** | Command used to add a new user profile to the list. |
| **Delete** | Command used to delete one or more selected user profiles from the list. If you delete a profile from the list, users with this profile will now have the default profile.<br>Preconfigured user profiles cannot be deleted from the list. |
| **Edit** | Command used to modify access rights associated with the selected user profile. |
| **Export** | Command used for exporting to a file the information (name and profiles) of the selected user profile(s). |
| **Import** | Command used for importing from a file the information (name and profiles) of the selected user profile(s). If a user profile is already on the list, a warning message is displayed. |

## Adding a user profile

Use the **Add** command to display the following dialog box:

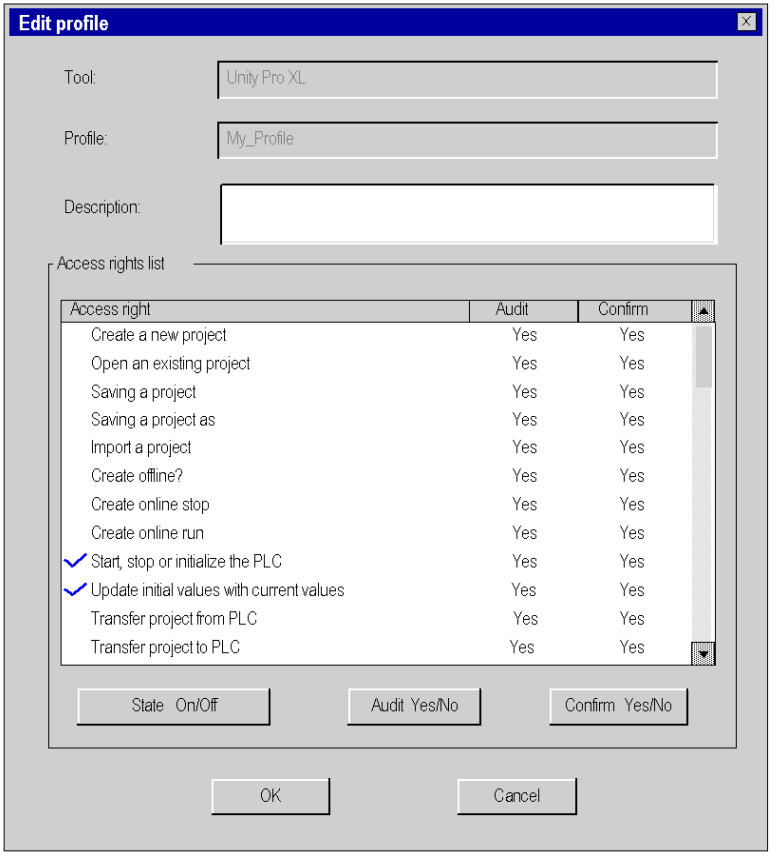



2 data entry fields are provided:

| Parameter | Description |
|---|---|
| **Add the profile** | This field is used to enter the name of the new user profile (2 to16 characters). If the name entered is not correct, or if it exists already, a warning message is displayed. |
| **With the following** | Field used to choose the user profile whose attributes you would like to retrieve. |

## Modifying a user profile

Use the **Edit** command to display the following dialog box:



4 areas are provided for display or data entry:

| Parameter | Description |
|---|---|
| **Tool** | Shows the name of the product that you selected in the previous screen. |
| **Profile** | Shows the user profile name that you selected in the previous screen. |
| **Description** | This data entry field is used to associate a comment with the user profile modification. |
| **Access rights list** | This list displays the access rights associated with the selected user profile.<br>● **Access right**: List of access rights associated with the user profile.<br>● **State On/Off**: This option is used to enable/disable a designated right for the current profile.<br>To enable/disable the right:<br>❍ Select **Access right** in the list.<br>❍ Click **State On/Off**: the selection mark appears/disappears.<br>● **Audit Yes/No**: When this option is enabled, it enables an operation to be stored in the log file.<br>To enable/disable the option for the corresponding access right:<br>❍ Select **Access right** in the list.<br>❍ Click **Audit Yes/No**.<br>● **Confirm Yes/No**: Enable this option to require a confirmation for an operation.<br>To enable/disable the option for the corresponding access right:<br>❍ Select **Access right** in the list.<br>❍ Click **Confirm Yes/No**. |

**NOTE:** You can view the access rights associated with a preconfigured profile, but you cannot modify them.

# Policies

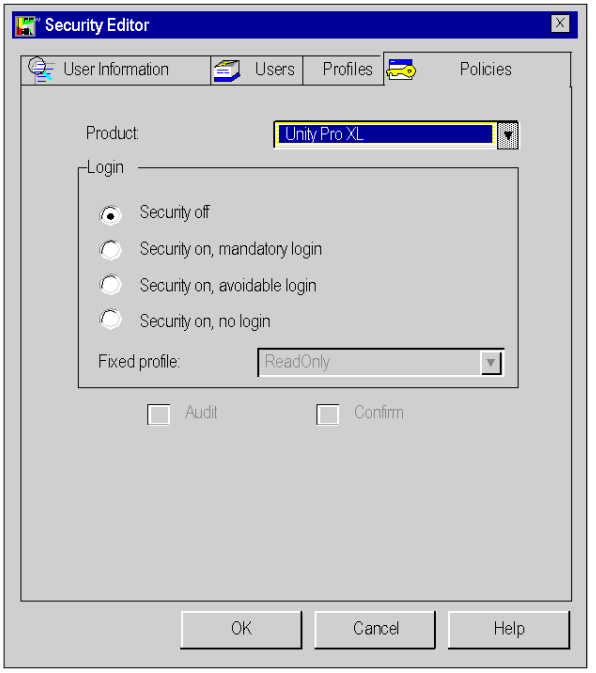
## At a Glance

This tab in the security editor can be accessed only by the super user. It enables the execution of the following functions:

- Determine the guidelines associated with a product,
- define the default profile,
- Enable/disable the **Check** option.
- Enable/disable the **Confirm** option.

## Policies

The **Policies** tab of the security editor looks as follows:



## Description of the parameters

The following table describes the screen parameters:

| Parameter | Description |
|---|---|
| **Product** | Used to choose the product for which you would like to define guidelines. In our case, Unity Pro. |
| **Login** | Used to define Unity Pro access rules:<br>• **Security off**: Security is disabled. You have direct access to Unity Pro. This is the default setting.<br>• **Security on, mandatory login**: Security is enabled. To access Unity Pro, it is mandatory that you enter your name and password, which will determine your profile.<br>• **Security on, avoidable login**: Security is enabled. To access Unity Pro, enter your name and password. In this case, you will have your user profile. You can also access Unity Pro without entering the password. In this case, you will have the default user profile.<br>• **Security on, no login** Security is enabled, but you have direct access to Unity Pro. The profile then is the default profile.<br>When the security is disabled, the **Audit** and **Confirm** options will also be disabled. |
| **Specified profile** | Used to define the default user profile. |
| **Audit** | Used to enable or disable the **Audit** option (this option is only available if security is enabled).<br>If this checkbox is enabled, a log file will be created to store Unity Pro user events.<br>The log file is located at **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Event Viewer** on your desktop. |
| **Confirm** | Used to enable or disable the **Confirm** option.<br>If this box is checked, enter your name and password each time secure access is defined for a user action *(see page 67)*.<br>**Note:**<br>If you are using the **Confirm** option in the **Policies** tab of the security editor, carefully select access rights when creating a new profile. Otherwise, changes will require multiple confirmations in the program sections. |

**NOTE:** The **Audit** and **Confirm** options are not assigned to one user but apply globally to all Unity Pro products.

# Recovery procedure with an error

## At a Glance

The security editor is used to control access to Unity Pro. If a problem occurs, (access attempted by an illegal user, loss of password, etc.), follow the recovery procedure corresponding to the problem.

## Access by an unknown user

If a user is not known to the security editor, and security is enabled, there are two options:

- Access is configured with an **avoidable login** (**Security On, Password Optional**): in this case, the unknown user can access Unity Pro with the default profile,
- Access is configured with **mandatory login** (**Security On, Password Required**): in this case, the unknown user cannot access Unity Pro.

If the database of the security editor is damaged or it was deleted, access to Unity Pro is not possible, even if security is not active (**Security Off**).

## Loss of password

If you have forgotten your password, the procedure to follow differs depending upon whether you are a user or a super user:

- if you are a user, contact the super user. He can reset your password. You can then enter a new password.
- If you are the super user, reinstall Unity Pro, choosing a customized installation: Install only the security editor.

## Database protection

To protect the database against possible damage, a backup file is created during installation. This hidden file can only be accessed in read mode. It enables the database to be restored when a fault occurs.

**NOTE:** The backup file is maintained and used if the database is damaged. If the automatic procedure fails, then repeat the installation procedure.

# Access rights

## At a Glance

Unity Pro access rights are classified in the following categories:

- project services
- adjustment/debugging
- libraries
- global modification
- elementary modification of a variable
- elementary modification of DDT compound data
- elementary modification of a DFB type
- elementary modification of a DFB instance
- bus configuration editor
- input/output configuration editor
- runtime screens

## Project services

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Create a new project** | You can create a new project. |
| **Open an existing project** | You can open an existing project. |
| **Save a project** | You can save the project. |
| **SaveAs a project** | You can copy the project. |
| **Import a project** | You can import a project. Partial import is considered to be a program modification. |
| **Build off-line** | You can launch generation of the executable in offline mode. |
| **Build on-line STOP** | You can launch generation of the executable in online mode, with the PLCs powered down. |
| **Build on-line RUN** | You can launch generation of the executable in online mode, with the PLCs powered up. |
| **Start, stop or initialize the PLC** | You can command the PLC (power up, power down, initialization). |
| **Update init values with current values** | You can copy the current values to update the initial values. |
| **Transfer project from PLC** | You can transfer the executable program from the PLC to the terminal. |
| **Transfer data init values from PLC** | You can transfer the initial values from the PLC. |
| **Transfer project to PLC** | You can transfer the executable program from the terminal to the PLC. |

| Access right | Description |
|---|---|
| **Transfer data values from file to PLC** | You can transfer data from a file to the PLC. |
| **Transfer data values from file to PLC** | You can transfer data from the PLC to a file. |
| **Restore project backup in PLC** | You can restore the contents of backup memory (Premium) or the memory card (Modicon M340 and Modicon M580) in the PLC executable area.<br>**NOTE:** On Modicon M580, you can save to flash memory if no memory card is inserted. |
| **Save to project backup in PLC** | You can save the project program in the backup memory (Premium) or the memory card (Modicon M340 and Modicon M580).<br>**NOTE:** On Modicon M580, you can restore from flash memory if no memory card is inserted. |
| **Set address** | You can access a PLC via the network. |
| **Modify options** | You can modify project attributes. |

### Adjustment/Debugging

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Modify variable values** | You can modify the value of the variables. |
| **Force internal bits** | You can force internal bits. |
| **Force outputs** | You can force outputs. |
| **Force inputs** | You can force inputs. |
| **Task management** | You can command execution of project program tasks (power up, power down; initialization). |
| **Task cycle time modification** | You can modify cycle time for cyclical tasks. |
| **Suppress message in viewer** | You can delete the message displayed in the viewer. |
| **Debug the executable** | You can debug the executable program. |
| **Replace a project variable** | You can replace all occurrences of one variable in the program by another. |

### Libraries

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Create libraries or families** | You can create libraries or families of user functions. |
| **Delete libraries or families** | You can delete libraries or families of user functions. |
| **Put an object into library** | You can insert an object in a library. |
| **Delete an object from library** | You can delete an object from a library. |
| **Get an object from a library** | You can import an object from a library to the project. |

### Global modification

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Modify the documentation** | You can modify the documentation. |
| **Modify the functional view** | You can create, delete, or modify a functional module. |
| **Modify the animation tables** | You can modify the structure of the animation tables (not the values). |
| **Modify constants value** | You can modify the value of the project constants. |
| **Modify the program structure** | You can modify the executable code structure (add/delete a section; modify the activation conditions for a section, modify section order.) |
| **Modify program sections** | You can modify the executable code for a section. |
| **Modify project settings** | You can modify the generation options. |

### Elementary modification of a variable

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Variable add/remove** | You can add or delete a variable. |
| **Variable main attributes modifications** | You can modify a variable's name, type, and address. |
| **Variable minor attributes modifications** | You can modify the comment and the initial value for a variable. |

### Elementary modification of DDT compound data

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **DDT add/remove** | You can add or delete a compound data item. |
| **DDT modifications** | You can modify the structure, comment and initial value for a compound data item. |

### Elementary modification of a DFB type

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **DFB type add/remove** | You can add or delete a DFB type. |
| **DFB type structure modification** | You can modify the structure of a DFB type. |
| **DFB type sections modification** | You can modify the code section of a DFB type. |

### Elementary modification of a DFB instance

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **DFB instance modification** | You can add, delete, or modify (name and type) a DFB instance. |
| **DFB instance minor attributes modification** | You can modify the comment and the initial value for a DFB instance. |

### Bus configuration editor

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Modify the configuration** | You can modify the configuration. |
| **I/O sniffing** | You can carry out bus configuration sniffing. |

### Input/output configuration editor

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Modify the I/O configuration** | You can modify the input/output configuration. |
| **Adjust the I/O** | You can adjust inputs/outputs. |
| **Save_param** | You can save the module input/output parameters explicitly (initial parameter values are replaced by their current values). |
| **Restore_param** | You can restore the module input/output parameters explicitly (initial parameter values replace their current values). |

### Runtime screens

The access rights for this category are as follows:

| Access right | Description |
|---|---|
| **Modify screens** | You can modify the runtime screens. |
| **Modify messages** | You can modify the runtime messages. |
| **Add/remove screens or families** | You can add or delete a runtime screen. |

# Section 4.2
## Project management

### Subject of this Section

This section introduces the tools used for project management.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Access to Unity Pro Software | 81 |
| Services in Offline Mode | 83 |
| Services in Online Mode | 92 |
| Allowed Online Modifications | 94 |
| Connection / disconnection | 99 |
| Accessing a PLC via a Network | 101 |
| Overview of Types of Connection Possible With Schneider PLCs | 107 |
| Accessing communication parameters | 111 |
| Project comparison | 114 |
| Conversion of Legacy Applications/Projects | 115 |
| Importing an LL984 Application | 117 |
| Transferring the project between the terminal and the PLC | 125 |
| Update of Upload Information | 129 |
| Sending a command to the PLC | 131 |
| Project Backup Management for Modicon M340 | 132 |
| Project backup for Premium | 134 |
| Save/Restore Data Between a File and the PLC | 136 |
| Compatibility Rules for Restore Using a *.DTX File | 140 |
| Transferring current values | 146 |
| Memory usage | 147 |
| Function Pack for Memory | 152 |
| Memory Card Access for Modicon M340 | 155 |
| FTP and Memory Card for Modicon M340 | 156 |
| Project Management with DTMs | 158 |

# Access to Unity Pro Software

## At a Glance

It is possible to launch several instances of Unity Pro simultaneously on the same workstation. However, the access method is identical for all instances and depends on whether the security option has been enabled:

- If security is active, clicking the Unity Pro icon calls up a login dialog box which is used to access Unity Pro.
- If security is inactive, you have direct access to Unity Pro.

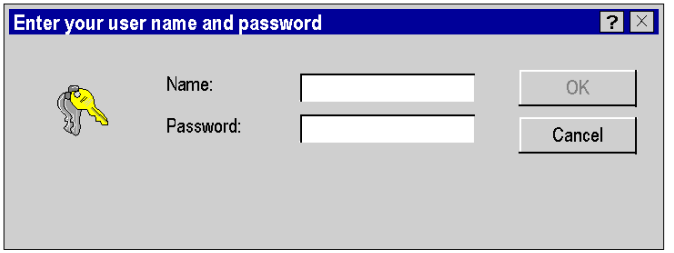
## Security Inactive or Active in No Login Mode

If security is inactive, execute the following commands to launch an instance of Unity Pro:

| Step | Action |
|---|---|
| 1 | Click **Start** → **Programs**. |
| 2 | Select the group **Schneider Electric** → **SoCollaborative**. |
| 3 | Select the group **Unity Pro**.<br>**Result**: A list appears<br>● Language selection<br>● OS Loader<br>● PLC Simulator<br>● Readme<br>● Registration Tool<br>● Types library update<br>● Unity Pro S, L, or XL |
| 4 | From the list, select **Unity Pro S, L, or XL**.<br>**Result**: Unity Pro is launched in the selected language.<br>Special case:<br>If you have not yet registered the software, a message warns you of the number of days of use left, and asks you whether you want to register (see Unity Pro installation manual). |

## Other Cases

When security is active *(see page 58)*, execute the following commands to launch an instance of Unity Pro:

| Step | Action |
|---|---|
| 1 | Activate the Unity Pro icon: see procedure below.<br>**Result**: The dialog box **Enter your user name and password** is displayed.<br>  |
| 2 | Enter your name and password (if necessary). |
| 3 | Confirm with **OK**. |

# Services in Offline Mode

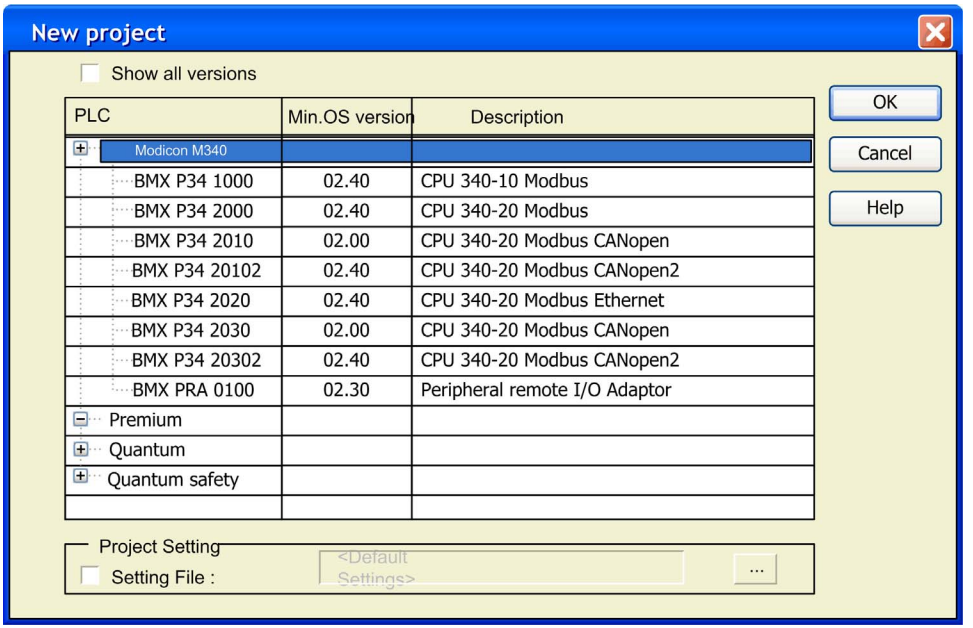
## At a Glance

The following operations can be carried out when accessing Unity Pro software in off-line mode:

- Creating,
- Opening,
- Converting,
- Analyzing,
- Generating,
- Saving,
- Closing.

## Creating a Project

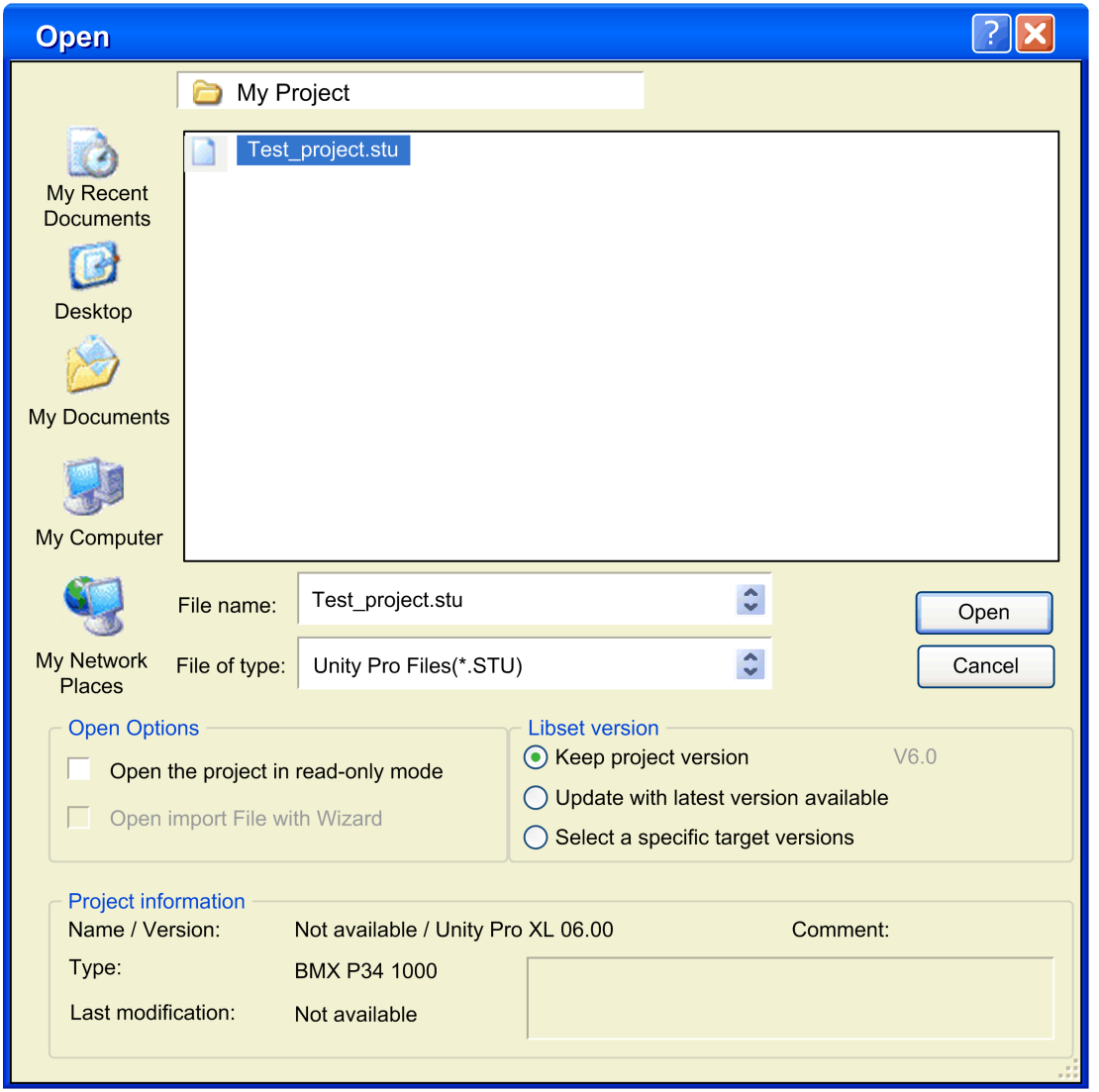
To create a new project, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Activate the **New** command in the **File** menu.<br>**Result**: The **New Project** screen appears.<br>  |
| 2 | To see all PLC versions, check **Show all versions** box. |

| Step | Action |
|---|---|
| 3 | Select the processor you wish to use from those proposed. |
| 4 | To create a project with specific values of project settings, check **Settings File** box and use the browser button to localize the **.XSO** file (Project Settings file). It is also possible to create a new one.<br>If the **Settings File** box is not checked, default values of project settings are used. |
| 5 | Confirm with **OK**. |

## Open Dialog

**Open** project dialog



| Element | Description |
|---|---|
| **Look in**<br>**File name**<br>**Files of type** | The upper part of this dialog displays a standard Windows **File Open** dialog. |
| Opening a legacy LL984 project | Please refer to Importing an LL984 Application *(see page 117)*. |
| **Open Options** | |
| **Open the project in read-only mode** | Opens the project in read-only mode. |
| **Open import File with Wizard** | Select this check box to use the import *(see page 1719)* wizard. |
| **Libset version**<br>**Libset** (set of libraries) contains all libraries, families, functions and variables (I/O data structures) that can be used to develop an automation project.<br>In general, to open an existing project (**.STU** file) or an archived project (**.STA** file) the Libset version can be selected.<br>To import a **.ZEF** or **.XEF** file, the latest Libset version must be selected.<br>More information on the default **Libset** version used to open a project according to the file type is provided in the libset *(see page 294)* section. | |
| **Keep project version** | Select this option to keep the **Libset** version used at file creation time.<br>If the respective **Libset** is not installed, the latest **Libset** version will be used. |
| **Update with latest version available** | Select this option to update the project with the latest **Libset** version. |
| **Select a specific target version** | Select this option to choose a specific **Libset** version. The **Version** box can be used to select a version. |
| **Project information** | |
| **Type** | Target PLC. |
| **Name / Version** | Project name and software package version. |
| **Last modification** | Date and time of last modification of the project. |
| **Comment** | Comment associated to the project. |

## Opening a Project

To open an existing project, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Choose the **Open** command from the **File** menu. |
| 2 | Select the file related to the project (file ***.STU**). |
| 3 | Select one of the **Libset version** options. |
| 4 | Confirm with **Open**. |

Opening a project also restores the Unity Pro context, which was saved when the project was closed. This involves:

- The list of open tools,
- For each tool: its window size, position, contents and animation status,
- taskbar and tool configuration.

The context file is a companion file to the <Project_name>.STU file. It is saved in the same directory with the name <Project_name>.ZTX. The **.ZTX** file is not required to open an application. If no **.ZTX** is provided the application is open with default context.

**NOTE:** When a project file is selected, the software provides a certain amount of information: the project name, associated comment, version and date the project was generated, target PLC, and date of last modification to the source code.

**NOTE:** The **.STU** file is a work file and it can not be used to switch from one version to another of Unity Pro software. In this case, you should use the archive format (**.STA** file) or use the export function in the project and create a **.ZEF** or **.XEF** file.

**NOTE:** The **.STU** file contains the configuration of the TSX CPP 110 in the **Project Browser**. A **The file not found** message appears each time the configuration screen of the TSX CPP 110 PCMCIA card is opened. Select the required **.CO** file to update the CANopen configuration.

## Opening an Archived Project

To open a project archived in STA format, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Choose the **Open** command from the **File** menu. |
| 2 | Select the file related to the project (file ***.STA**). |
| 3 | Select one of the **Libset version** options. |
| 4 | Confirm with **Open**. |

An STA file is a project archive file that can be accessed using the **Archive** command in the **File** menu.

The properties of the **.STA** file are the following:

- The **.STA** file is very compressed (approximately 50 times more so than the STU). It is used to transfer projects to networks (local or Internet, for example).
- The **.STA** file can be used to transfer projects between versions of Unity Pro software (for example, between version 2.0 and version 4.1).
- The **.STA** file contains the entire project:
  - The PLC binary,
  - The Upload information: comments and animation tables,
  - The operator screens.

**NOTE:**
When a **.STA** file is selected, the software provides a certain amount of information:

- The project name,
- Associated comment,
- Version and date the project was generated,
- Target PLC,
- Date of last modification to the source code,
- The version of Unity Pro used to create this archive.

**NOTE:** As the file is compressed, loading takes much longer than for an **.STU** file. We recommend using the **.STA** file to archive the project and the **.STU** file as the actual work file.

**NOTE:** To Fully retrieve an application with Quantum Ethernet I/O system and/or 140 NOC 78• 00 topologies, The **Upload information** or **Open** an archive files must be used.

## Converting an Existing PL7 or Concept Project

To convert an existing PL7 or Concept project, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Choose the **Open** command from the **File** menu. |
| 2 | Select the file related to the project:<br>● ***.FEF** or ***.DFB**, to launch the conversion procedure for a PL7 application *(see Unity Pro, PL7 Application Converter, User Manual)*,<br>● ***.ASC**, to launch the conversion procedure for a Concept application *(see Unity Pro, Concept Application Converter, User Manual)*. |
| 3 | Confirm with **Open**. |

## Conversion Wizard

To convert a legacy project partially or as a whole the **Conversion Wizard** can be used.

The **Conversion Wizard** provides options to remap I/O objects (channels, variables etc.) during conversion and to adapt the hardware configuration of the new project concurrently in Unity Pro.

## Shortcuts to Recent Projects

To simplify access to the current project(s), the software provides a shortcut for accessing the last 8 projects used. In the File menu, activate the shortcut to the project you would like to open.

## Archiving a Project

To archive a project in STA format, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the **Save Archive** command from the **File** menu. |
| 2 | If necessary, select the directory where the project will be stored (disk and path). |
| 3 | Enter the file name. |
| 4 | Confirm with **Save**. |

**NOTE:** Archiving is only possible after a **Rebuild all project** with the Upload information *(see page 519)* section of the **Project Settings** window set as follows:

- the **Upload information** and **Comments** check boxes are checked, with the **Animation tables** check box unchecked
- the **Upload information**, **Comments** and **Animation tables** check boxes are checked
- the **Upload information**, **Comments** and **Animation tables** check boxes are unchecked

**NOTE:** The .STA file partially contains the TSX CPP 110 configuration. Do not forget to backup the **.CO** file (Sycon) for CANopen after archiving of the project.

## Analyzing a Project

To analyze the syntax of your project, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Activate the **Analyze project** command in the **Build** menu.<br>**Result**: the software analyzes the project. |
| 2 | Any errors detected are displayed in the information window at the bottom of your screen. |

## Analyzing and Generating a Project

To analyze and generate your project at the same time (generation of links between input/output modules and declared objects in the project, etc.), carry out the following actions:

| Step | Action |
|---|---|
| 1 | Activate the **Rebuild all Project** command in the **Build** menu.<br>**Result**: the software analyzes and generates the project. |
| 2 | Any errors detected are displayed in the information window at the bottom of your screen. |

## Analyzing and Generating Modified Project Parts

To analyze and generate **only** the modified parts of your project, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Activate the **Build Changes** command in the **Build** menu.<br>**Result**: the software analyzes and generates the modified parts of your project. |
| 2 | Any errors detected are displayed in the information window at the bottom of your screen. |

**NOTE:** The **Build Changes** analyses and displays only modified sections or sections impacted by modifications. To display all warnings, a **Rebuild all Project** command is needed.

**NOTE:** The **Build Changes** command can be executed only if the **Rebuild all Project** command has been executed at least once previously.

### Backing up a New Project

To back up a new project, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Activate the **Save** or **Save As** command in the **File** menu. |
| 2 | If necessary, select the directory where the project will be stored (disk and path). |
| 3 | Enter the file name. |
| 4 | Confirm with **Save**. |

### Backing up an Existing Project

To back up your modifications, select the Save command from the File menu.

### Closing a Project

To close your project, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Activate the **Close** command in the **File** menu. |
| 2 | If you have modified your project, a dialog box is displayed with the message that you must save your project. |
| 3 | Save your project or close it without saving changes. |

**NOTE:** Closing a project triggers saving the current context in the file <Project_name>.ZTX *(see page 87)*.

### Exiting Unity Pro

To exit Unity Pro, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Activate the **Exit** command in the **File** menu. |
| 2 | If you have modified your project, a dialog box is displayed with the message that you must save your project. |
| 3 | Save your project or exit it without saving changes. |

# Services in Online Mode

## Programming and monitoring modes

You can access the project in the PLC in Online Mode in the following two ways:

- **Programming**, the PLC is reserved by the terminal; it is possible to access all the functions of the software. It is not possible to connect in programming mode to a PLC that is already reserved.
- **Monitoring**, The PLC is not reserved by the terminal; it is not possible to modify functions of the PLC, (the program), however it is possible to modify values of the variable with the animation table.It is also possible to connect in monitoring mode to a PLC that is already reserved. Links between EFs are not animated.

**Note:** The command **Tools → Options...Connection** tab lets you choose the default operating mode *(see page 561)*.

## Equal or Different Modes

On connection, the data in the PLC and the terminal is compared *(see page 519)*:

- if the data is identical, this determines an equal online mode *(see page 99)*,
- if the data is different, this determines a different online mode *(see page 99)*.

## Information in the PLC

Two types of information are compared between the PLC and the terminal:

- Executable information is used to identify and execute the project in the PLC:
  - identification of the project,
  - configuration,
  - code sections,
  - DDT types,
  - DFB types,
  - EF and EFB codes,
  - project options.
- Upload information:
  - project upload information: graphic code for languages with LD contacts and chart of FBD function blocks, symbols of static and dynamic variables,
  - comments: variables, code sections, project, DFBs, etc.,
  - animation tables.

## User functions

The user functions provided in online mode that can be accessed from the **PLC** menu are the following:

| Function | Description |
|---|---|
| **Connection/Disconnection** | Connection (change to online mode) or disconnection (change to offline mode). |
| **Set Address** | Access to a PLC via the network. |
| **Standard mode** | Choice of connection with a PLC. |
| **Simulator mode** | Choice of connection with a PLC. |
| **Compare** | Comparison of terminal/PLC projects. |
| **Transfer Project to PLC** and **Transfer Project from PLC** | Transfer of the program between the terminal and the PLC. |
| **Save Data from PLC to File** and **Restore Data from File to PLC** | Transfer of data between a file and the PLC. |
| **Run/Stop** and **Init** | Sending a command to the PLC. |
| **Update Upload Information** | Update the Upload information in the PLC. |
| **Update Init Values with Current Values** | The PLC forces the update of the initial values using current values (this applies to the variables with the save attribute). |
| **Project Backup** | Backs up the project in memory. |
| **Memory Consumption** | Access to PLC memory usage statistics. |
| **Diagnostics** | Access to the debug function. |

## Cancel/Repeat

The **Cancel/Repeat** commands in the **Edit** menu are not available in the following cases:

- Deletion of a step or macro-step in online mode.
- Switch from **normal view** to **extended view** in SFC debug mode and vice-versa.

# Allowed Online Modifications

## Overview

The following pages describe modifications that are allowed when Unity mode is online.

## General Rules Concerning Online Modifications

Online modification is a feature that concerns only the ‘online equal’ state.

If the application present in the PLC is different from the one opened in Unity Pro, the state is ‘online different’. One of both applications should be chosen and loaded to be in the ‘online equal’ state.

**NOTE:** To download the application from Unity Pro into the PLC, it is mandatory to stop the PLC. However, the PLC may stay in RUN mode while uploading the application to the PC via Unity Pro.

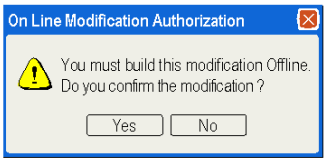
The ‘online equal’ state may only be reached if the connection mode is ‘programming’. To configure the connection mode, use the command **Tool** → **Options** → **Connection**.

Two different connection modes are available:
- monitoring *(see page 92)*: state is ‘monitoring’ and no online modification is allowed,
- programming *(see page 92)*.

The distinction between executable code and upload information means that there are 2 kinds of modifications:
- those which truly modify the executable information (e.g., which add a new EFB call or a new ST instruction)
- modifications that have an impact only on the Upload information (e.g. concerning comments, graphical information of languages).

Consequently, these kinds of modifications may not need to be built depending on whether upload information is loaded into PLC, comments and animation tables included in upload information.

As a general rule, Unity Pro prevents the user from doing actions by mistake that would force them to stop the PLC and/or to perform a complete download:
- either by simply disabling the forbidden functions in online mode
- or by a warning message, as follows:




To know what parts have been modified, the application browser displays a specific mark for the modified parts and in the title bar of the editors by character ‘*’.

## Modify the Program

It is always possible to modify a section of the program in online mode. As soon as a section has been modified, it is no longer animated until the next build and transfer to PLC.

Nevertheless, depending on the kind of modifications, a SFC graph may be reset after the build and PLC update.

Some modifications performed in SFC might need a **rebuild all** command. In such cases, a warning message appears in the editor. For example, deleting an action or a transition section requires a 'rebuild all' command. However, removing an action or a transition section from a step or a transition and keeping the section unused only requires an online build.

**NOTE:** Adding an SFC section must be done offline, if the two following conditions are fulfilled simultaneously:

- If the project contains at least one SFC section already.
- If the project has been designed with an earlier version than Unity Pro 3.0.

This modification requires a **rebuild all** of the project and a PLC stop to download the entire application.

## Modify Data

In the data editor, there is no specific operating mode.

Generally, it is allowed to modify the type of a variable or a Function Block (FB) in online mode.

Modifying a data, which is used in the program, may have consequences wherever it is used, on the contrary to an unused variable. Therefore online actions are more restricted on used data.

**NOTE:** Adding a variable without rebuilding increases the memory usage until next rebuild.

For used EDT variable or used DDT/FB instance, only comments and the initial values can be modified. Unused variable/instance may be deleted. It is allowed to create new variable/instance online.

Concerning DDT type, it is only allowed to change comments if the type is used. It is possible to create new DDT or to delete an unused one.

For DFB type, it is possible to:

- create, modify and delete sections
- change comments
- change initial values
- add a new public or private variable

If there is no instance of the DFB, there is no limitation. The DFB type may even be deleted. Finally creating new DFB type(s) and instantiating it in online mode is straightforward.

## Modify the Configuration

For Premium and Atrium PLCs, it is not possible to add or remove any module in online mode. Parameters of an existing module on the contrary may be configured online. Once modified, the animated mode is no longer available until the next build and PLC update.

For Quantum system PLCs, it is possible to add or delete a selection of modules and change the parameters of an existing module *(see Modicon Quantum, Change Configuration On The Fly, User Guide)* in STOP and RUN mode.

## Comment Concerning Offline Modifications

Modifications that are only possible offline and new application downloads require an offline build and a rebuild all. In these cases, the modification is not possible online and a message notifies the user.

The rebuild is needed:

- After application creation or import
- Once the application is already built:
  - Changing some project settings
  - CPU change
  - Memory card change
  - Memory card layout change
  - Switching from real target to simulator and vice versa

Furthermore, if a flash PCMCIA card without additional SRAM (TSX MFPP •••M or TSX MFPP •••K) is configured, no modification is allowed online. User is informed by a specific message ‘This action is forbidden because: Flash Cartridge’.

## Synthesis of Authorized and Unauthorized Modifications

The following table presents the main authorized and unauthorized online modifications. The second column indicates which modifications have impact only to the upload information and do not have any impact to the executable code:

| Type of Modification | Impacts Only Upload Information | Allowed in Online Mode (RUN/STOP) | Requires to Build the Application in Offline Mode |
|---|---|---|---|
| **General** | | | |
| Name of Station, Programs, Sections | Yes | Yes | No |
| Comment of Station, Configuration, Programs, Sections | Yes | Yes | No |
| Documentation Summary | Yes | Yes | No |
| Animation Table | Yes | Yes | No |

**(1)** Modifying only the graphical layout or the free form comments does not impact code generation, but only generation of the upload information.
**(2)** This is only available in the RUN mode for the Quantum modules.

| Type of Modification | Impacts Only Upload Information | Allowed in Online Mode (RUN/STOP) | Requires to Build the Application in Offline Mode |
|---|---|---|---|
| Integrated Operator Screens | Yes | Yes | No |
| Functional View | Yes | Yes | No |
| Security Information: Passwords, Protection Attributes,... | No | Yes | No |
| Switch Simulator <-> Real Target | No | No | Yes |
| Import/Export Project | No | Yes | No |
| **Program** | | | |
| Sections of Programs: Add, Change Execution Order | No | Yes | No |
| Modify the Code of a Section (Task Section, SR, Transition, Action, DFB Sections) | Yes(1) | Yes | No |
| Modify the Code of EVT Sections | No | Yes(2) | Yes |
| Modify the Code of SFC Chart | No | Yes | No(1) |
| **Configuration/Communication** | | | |
| Premium Configuration: Add/Delete a Module | No | No | Yes |
| Premium Configuration: Change Module Parameters | No | Yes | No |
| Quantum Configuration: Add/Delete a Module | No | Yes | No |
| Quantum Configuration: Change Module Parameters | No | Yes | No |
| Changing Memory Sizes in Configuration Screen | No | No | Yes |
| **Global Variables (Used Even in Animation Table or Operator Screen)** | | | |
| Remove an Used Variable | No | No | Yes |
| Symbol an Used Variable | No | Yes | No |
| Type an Used Variable | No | No | Yes |
| Topological Address of an Used Variable | No | Yes | No |

**(1)** Modifying only the graphical layout or the free form comments does not impact code generation, but only generation of the upload information.
**(2)** This is only available in the RUN mode for the Quantum modules.

| Type of Modification | Impacts Only Upload Information | Allowed in Online Mode (RUN/STOP) | Requires to Build the Application in Offline Mode |
|---|---|---|---|
| Initial Value of an Used Variable | No | Yes | No |
| Comment of an Used Variable | Yes | Yes | No |
| Create, Remove or Modify Unused Variables (EDT, DDT) | No | Yes | No |
| Create, Remove or Modify Unused Variables (FB) | No | Yes | No |
| **Used DFB** | | | |
| Type Name of Used DFB | No | No | Yes |
| All Comments | Yes | Yes | No |
| Add a Parameter | No | No | Yes |
| Add a Public or Private Variable | No | Yes | No |
| Delete or Change Unused Private Variable | No | Yes | No |
| Initial Value of Parameters and Variables | No | Yes | No |
| Other Attributes of Parameters and Variables (Public and Private) | No | No | Yes |
| Modify the Code of a section | No | Yes | No |
| Create a New DFB Type | No | Yes | No |
| Delete an Unused DFB Type | No | Yes | No |
| **Used DDT** | | | |
| Create a New DDT Type | No | Yes | No |
| Delete an Unused DDT Type | No | Yes | No |

**(1)** Modifying only the graphical layout or the free form comments does not impact code generation, but only generation of the upload information.
**(2)** This is only available in the RUN mode for the Quantum modules.

# Connection / disconnection

## At a Glance

Connection enables you to change from offline to online mode. On connection, data in the PLC and the terminal is compared:

- if the information is identical, the online mode is Equal,
- if the information is different, the online mode is Different,

By disconnecting, you can return to offline mode from an online mode.

## Equal online mode

On connection to the PLC, Unity Pro runs in Equal online mode when executable files and the Upload information are identical in the PLC and in the terminal.

You can modify the project in the same way as for offline mode. Generating the executable also automatically triggers loading of modifications to the PLC.

If the project program has been modified in the terminal without regenerating the executable, only the non-modified parts of the project will be animated.

To animate the entire project once again, you must generate the executable and update the PLC.

## Different online mode

On connection to the PLC, Unity Pro runs in Different online mode when executable files and the Upload information are different in the PLC and in the terminal.

To run in Equal online mode, you must transfer the project between the terminal and the PLC.

In Different online mode, the animation level is limited to access to located variables via their topological address. Modification of the program is not authorized.

## Connection procedure

You must execute the following actions to connect up:

| Step | Action |
|---|---|
| 1 | On first connection: Click *(see page 101)* **PLC** → **Set Address**. |
| 2 | Choose the connection mode:<br>● **PLC** → **Standard mode**: to connect to a PLC<br>● **PLC** → **Simulator mode**: to connect to a simulator |
| 3 | Click **PLC** → **Connection**. |

## Disconnection procedure

You must execute the following actions to disconnect:

| Step | Action |
|---|---|
| 1 | Click **PLC** → **Disconnection**. |

## Automatic disconnection

In order to avoid online mode switching automatically to offline mode, without your explicit request (which is possible if the process executed by the processor is too large), you are recommended to increase the Timeout communication parameter *(see page 111)*.

# Accessing a PLC via a Network

## Procedure

You must carry out the following actions to access a new PLC via the network:

| Step | Action |
|---|---|
| 1 | Activate the menu **PLC** → **Set address** command, which displays the **Set address** dialogue. |
| 2 | In the **PLC Address** field, enter the address of the new PLC. |
| 3 | In the **PLC Media** field, select the type of communication you wish to use *(see page 107)*. For Ethway drivers, refer to the following note. |
| 4 | If you wish, modify:<br>● the communication parameters *(see page 111)* using the **Communication Parameters** buttons in the **PLC** and **Simulator** group boxes<br>● the connection bandwidths *(see page 112)* using the **Bandwidth** button |
| 5 | To carry out a connection test click on the **Test Connection** button. |
| 6 | When you are finished, confirm with the **OK** button. |

**NOTE:** In the **Set Address** dialogue, Ethway drivers are not available. To use this type of communication, you must enter manually ETHWAY01 or ETHWAY02 in the **Media** drop-down menu.

**NOTE:** By default the simulator is started on port number 502. To change the port number (portnum), add in the **Simulator Address** box an unused port number (e.g., 127.0.0.1:511 to use port number 511).

## Description of Set Address Parameters

The following table describes the **Set Address** dialogue:

| Group box | Parameter | Description |
|---|---|---|
| **PLC** | **Address** | PLC address (by default, the address of the PCL physically wired to the terminal).<br>The drop-down menu lists any previous addresses entered. Clicking on the trashcan clears the drop-down list |
| | **Media** | This drop-down selects the type of communication with the network for the PCL. |
| | **Communication Parameters** | By clicking this button you access the PLC communication parameters *(see page 111)*. |
| **Simulator** | **Address** | Simulator address<br>The drop-down menu lists any previous addresses entered. Clicking on the trashcan clears the drop-down list |
| | **Media** | This drop-down selects the type of communication with the network for the simulator. |
| | **Communication Parameters** | By clicking this button you access the simulator communication parameters *(see page 112)*. |
| | **Bandwidth** | Click this button to display the **Online Mode Services Bandwidth** dialogue *(see page 112)*. |
| | **Test Connection** | Click this button to execute a connection test without closing the **Set Address** dialogue. |
| | **Speed rate auto-adaptation at the end of download** | Choose this mode for applications which do not require a specific speed rate (e.g., Modems). |

## Network Address Syntax

The network address syntax is documented in the Architecture and Communication Services *(see Communication Services and Architectures, Reference Manual)* manual.

The following table describes the various authorized addresses according to the platforms and media supports.

| Platform | Support | | Address syntax |
|---|---|---|---|
| Premium | Uni-Telway | SCP01 | X-Way |
| | Uni-Telway | UNTLW01 | X-Way Example: UNTLW01 0.2.1.4 access to the Uni-Telway slave at address 4 which is connected to rack 0, module 2, channel 1 of the local PLC. |
| | Fipway | FIP01, FIP02, FPP2001 (driver for the TSX FPP20 card) | X-Way |
| | Fipio | FIP01, FIP02, FPP2001 | X-Way Example: FIP02 \2.55\SYS to access to FIPIO connection point 55 through the privileged terminal Fipio (address 63 for Premium)<br>Example: FIP01 SYS to access to Fipio Master through the privileged terminal Fipio (address 63 for Premium). |
| | PCIway | PCIWAY01, PCIWAY02 | X-Way |
| | Ethway | ETHWAY01, ETHWAY02 | X-Way |
| | XIP | XIP01 to XIP05 | X-Way Example: XIP01 {3.5}\2.55\SYS to access a FIPIO connection point 55, through the network 3 and station 5 PLC.<br>Example: XIP02 {3.5}0.2.0.4 access to a Uni-Telway slave connection at address 4 which is connected on the rack 0 module 2 channel 0 of the network 3 and station 5 PLC. |
| | Ethernet IP | TCP/IP | IP address |
| | Ethernet IP | TCP/IP** | IP address;Index or<br>"DNS machine name;Index" |
| | USB | USB | implicit; no address is necessary as the connection is point to point. However, it is possible to enter an X-Way address when the PLC is acting as an X-Way router |
| | Modbus Plus | MBPLUS01 to MBPLUS04 | Modbus Plus Address Example: MBPLUS01 2.1 access to Modbus Plus node address 1 which is connected after the Modus Bridge address 2. |

| Platform | Support | | Address syntax |
|---|---|---|---|
| Quantum | Serial Modbus | MODBUS01 to MODBUS04 | Modbus Slave Number |
| | Modbus Plus | MBPLUS01 to MBPLUS04 | Modbus Plus Address |
| | Ethernet | TCP/IP | IP Address Example: TCPIP 84.0.127.6 access to an Ethernet TCP/IP station. |
| | Ethernet IP | TCP/IP*** | IP address;Index or<br>"DNS machine name;Index" |
| | USB | USB | implicit; no address is necessary as the connection is point to point |
| Modicon M340 | Serial Modbus | MODBUS01 to MODBUS04 | Direct Address: Modbus slave number<br>Remote address: Link_address.Modbus Slave number<br>Link_address = r.m.c. of Modbus link<br>Direct address example: 66<br>Bridging example:<br>● 66\\0.0.3{118.159.35.45} for Modbus slave 66 to PLC on Ethernet CPU.<br>● 66\\0.3.0{118.159.35.45} for Modbus slave 66 to PLC on Ethernet NOE on slot 3. |
| | Ethernet | TCP/IP | Direct address: IP Address<br>Remote address: Netlink {IP address}<br>Netlink address = r.m.c. of Ethernet link<br>Direct address example: 84.0.127.6<br>Bridging example:<br>● 139.160.235.20\\0.3.0{118.159.35.45} for Ethernet CPU to PLC on Ethernet NOE on slot 3.<br>● 139.159.35.20\\0.0.0.66 for Ethernet to Modbus slave 66. |
| | Ethernet IP | TCP/IP** | IP address;Index or<br>"DNS machine name;Index" |
| | USB | USB | SYS or empty<br>Bridging example: SYS\\0.0.0.66 for USB to Modbus slave 66. |

| Platform | Support | | Address syntax |
|---|---|---|---|
| Modicon M580 | Ethernet | TCP/IP | Direct address: IP Address<br>Remote address: Netlink {IP address}<br>Netlink address = r.m.c. of Ethernet link<br>Direct address example: 84.0.127.6<br>Bridging example:<br>● 139.160.235.20\\0.3.0{118.159.35.45} for Ethernet CPU to PLC on Ethernet BMX NOR on slot 3.<br>● 139.159.35.20\\0.4.0.66 for Ethernet to Modbus slave 66 of channel 0 of a BMX NOM 0200 on slot 4. |
| | Ethernet IP | TCP/IP** | IP address;Index or<br>"DNS machine name;Index" |
| | USB | USB | SYS or empty<br>Bridging example: SYS\\0.4.0.66 for USB to Modbus slave 66 of channel 0 of a BMX NOM 0200 on slot 4. |
| Simulator | Ethernet | TCP/IP | IP address;Index **<br>"localhost"<br>"DNS machine name;Index" ** |
| Legend | **: "Index" is the Modbus Plus or Modbus index stipulated in the bridge you want to cross. This access only works in display mode: it cannot be used to modify the program or force values.<br>***: "Index" is the Modbus Plus or Modbus index stipulated in the bridge you want to cross.<br>**Note**: You must use quotation marks when the address includes characters other than figures, full stops or semi colons. These are used to avoid this address being analyzed syntactically. | | |

## Accessing PLCs Behind a Router

When you need to communicate to several PLCs behind routers, configure a network address translation table in Unity Pro to handle different ports. You supply the router address and different ports to reach PLCs behind the routers.

**Example**:

Suppose the router address is 139.160.39.12. The PLC addresses below are local:

| PLC Number | PLC Address |
|---|---|
| 1 | 192.168.0.1:502 |
| 2 | 192.168.0.2:502 |
| 3 | 192.168.0.3:502 |
| This example shows PLCs with different addresses, but the same port. | |

### Network Address Translation Table

The network address translation table would look something like this:

| Port | Destination |
|---|---|
| TCP/IP 3501 | 192.168.0.1:502 |
| TCP/IP 3502 | 192.168.0.2:502 |
| TCP/IP 3503 | 192.168.0.3:502 |
| This example shows the same router address with different ports, each port corresponding to a PLC address. | |

**Notes:**

- Unity Pro can communicate with PLC 2 using the router address followed by port 3502.
- In the **Set Address** dialog, type 139.160.39.12:3502.
- The router sends the message to the address 192.168.0.2:502 (configured in the router in the network address translation table).
- To communicate with other PLCs, use the same router address, but change the port number.

# Overview of Types of Connection Possible With Schneider PLCs

## At a Glance

Unity Pro connects to Modicon M340, Modicon M580, Premium, Atrium, and Quantum PLCs in different ways.

Each type of connection has its own characteristics. The following table provides an overview of the different possibilities.

## Connection Solutions

| ⚠ CAUTION |
| --- |
| **UNEXPECTED APPLICATION BEHAVIOR**<br>Do not connect more than one PLC over USB on the same computer.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

The following table shows the different solutions that can be used to connect a PLC with Unity Pro software.

| Type of Connection | M340 | M580 | Premium and Atrium | Quantum |
|---|---|---|---|---|
| USB | All processors | All processors | On processors:<br>● TSX P57 4634<br>● TSX P57 554<br>● TSX P57 5634<br>● TSX P57 6634 | On processors:<br>● 140 CPU 651 50<br>● 140 CPU 651 60<br>● 140 CPU 651 60S<br>● 140 CPU 652 60<br>● 140 CPU 658 60<br>● 140 CPU 670 60<br>● 140 CPU 671 60<br>● 140 CPU 672 60<br>● 140 CPU 672 61<br>● 140 CPU 671 60S<br>● 140 CPU 678 61 |
| Uni-Telway | - | - | All processors, via terminal port and PCMCIA cards | - |
| Modbus (1) | On processors:<br>● BMX P34 1000<br>● BMX P34 2000<br>● BMX P34 2010<br>● BMX P34 20102<br>● BMX P34 2020 | - | - | All processors have a Modbus connection which can be used by Unity Pro. |

**(1)** Bluetooth connection is limited to Diagnostics and Animation.
**(2)** Requires a valid application to be installed on the PLC and requires the communication cards or modules used to be configured correctly.

**NOTE:**

For Modicon M580 and M340 PLCs, the TCP/IP connection does not require a valid application inside the PLC to work.

- For details on Modicon M580 default address configuration, refer to *Unity Pro Configuration Tabs (see Modicon M580, Hardware, Reference Manual)*.
- For details on Modicon M340 default address configuration, refer to *Methods for IP Addressing (see Modicon M340 for Ethernet, Communications Modules and Processors, User Manual)*.

**(3)** Cannot be used to view application-specific screens in online mode.
**(4)** Programming mode *(see page 92)* cannot be accessed via a Modbus Plus network.

| Type of Connection | M340 | M580 | Premium and Atrium | Quantum |
|---|---|---|---|---|
| XIP (2) | - | - | Via the following modules:<br>● TSX ETY 110<br>● TSX ETY 4103<br>● TSX ETY 5103<br>● TSX ETY PORT<br>● Ethernet channel of TSX P57 4634<br>● Ethernet channel of TSX P57 5634<br>● Ethernet channel of TSX P57 6634 | - |
| Modbus Plus | - | - | Via PCMCIA cards<br>TSX MBP 100 (2)(3)(4) | All processors |
| Ethernet TCP/IP (2) | Processors Ethernet channels:<br>● BMX P34 2020<br>● BMX P34 2030<br>● BMX P34 20302<br>Via the following modules:<br>● BMX NOE 0100<br>● BMX NOE 0110 | All processors | Processors Ethernet channels:<br>● TSX P57 4634<br>● TSX P57 5634<br>● TSX P57 6634<br>Via the following modules:<br>● TSX ETY 4103 (3)<br>● TSX ETY 5103 (3)<br>● TSX ETY PORT (3) | Processors Ethernet channels:<br>● 140 CPU 651 50<br>● 140 CPU 651 60<br>● 140 CPU 651 60S<br>● 140 CPU 652 60<br>● 140 CPU 658 60<br>Via the following modules:<br>● 140 NOE 211 x0<br>● 140 NOE 251 x0<br>● 140 NOE 771 00<br>● 140 NOE 771 01<br>● 140 NOE 771 10<br>● 140 NOE 771 11<br>● 140 NOE 771 20 |

**(1)** Bluetooth connection is limited to Diagnostics and Animation.

**(2)** Requires a valid application to be installed on the PLC and requires the communication cards or modules used to be configured correctly.

**NOTE:**

For Modicon M580 and M340 PLCs, the TCP/IP connection does not require a valid application inside the PLC to work.

- For details on Modicon M580 default address configuration, refer to *Unity Pro Configuration Tabs (see Modicon M580, Hardware, Reference Manual)*.
- For details on Modicon M340 default address configuration, refer to *Methods for IP Addressing (see Modicon M340 for Ethernet, Communications Modules and Processors, User Manual)*.

**(3)** Cannot be used to view application-specific screens in online mode.

**(4)** Programming mode *(see page 92)* cannot be accessed via a Modbus Plus network.

| Type of Connection | M340 | M580 | Premium and Atrium | Quantum |
|---|---|---|---|---|
| PCI | - | - | On Atrium processors only:<br>● TSX PCI 57 204<br>● TSX PCI 57 354 | - |
| Fipio (2) | - | - | All PLCs connected to the Fipio bus to which the PC running Unity Pro is connected, as well as all those accessible via X-Way routers | - |
| Fipway (2) | - | - | All PLCs connected to the Fipway network to which the PC running Unity Pro is connected, as well as all those accessible via X-Way routers | - |

**(1)** Bluetooth connection is limited to Diagnostics and Animation.
**(2)** Requires a valid application to be installed on the PLC and requires the communication cards or modules used to be configured correctly.

**NOTE:**

For Modicon M580 and M340 PLCs, the TCP/IP connection does not require a valid application inside the PLC to work.

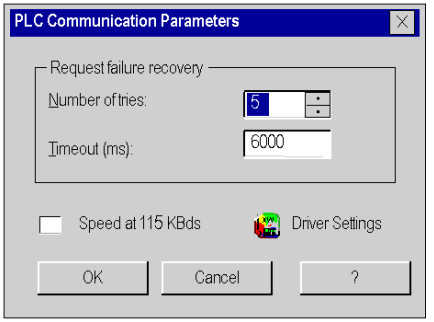
- For details on Modicon M580 default address configuration, refer to *Unity Pro Configuration Tabs (see Modicon M580, Hardware, Reference Manual)*.
- For details on Modicon M340 default address configuration, refer to *Methods for IP Addressing (see Modicon M340 for Ethernet, Communications Modules and Processors, User Manual)*.

**(3)** Cannot be used to view application-specific screens in online mode.
**(4)** Programming mode *(see page 92)* cannot be accessed via a Modbus Plus network.

# Accessing communication parameters

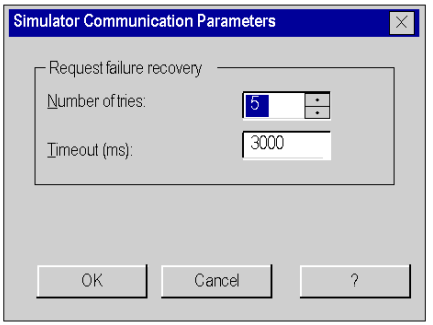
## Procedure for accessing PLC parameters

You must perform the following actions to access the communication parameters of the PLC:

| Step | Action |
|---|---|
| 1 | Access the **Set Address** screen *(see page 101)*. |
| 2 | Click on the **Communication Parameters** button in the **PLC** field to display the following screen:<br>  |
| 3 | Modify the parameters you wish to change:<br>● Enter the number of attempts you wish to be made for failure recovery in the **Number of tries** field.<br>● Enter the timeout delay to be used for failure recovery in the **Timeout (ms)** field:<br>❍ For Uni-Telway, the recommended value is: 4000 ms.<br>❍ For Modbus ASCII, the recommended value is: 6000 ms.<br>● Check the **Speed at 115 KBds** box to increase the project loading speed: from the configured speed to 115 KBds. This option is only available in **Uni-Telway** mode for **TSX 57 3••** and **TSX 57 4••** processors. For all other types of processor the Speed at 115 KBds option has no effect.<br>**Note**: Click on **Driver Settings** to display the driver management screen *(see Communication Drivers, Installation Manual)*. |
| 4 | Confirm with **OK**.<br>**Note**: Any modification to the communication parameters of the PLC will only actually be taken into account if it is confirmed by clicking **OK** in the **Set Address** screen. Clicking **Cancel** cancels acknowledgment of any modifications. |

## Procedure for accessing simulator parameters

You must perform the following actions to access the communication parameters of the simulator: *(see Unity Pro, Program Languages and Structure, Reference Manual )*

| Step | Action |
|---|---|
| 1 | Access the **Set Address** screen *(see page 101)*. |
| 2 | Click on the **Communication Parameters** button in the **Simulator** field to display the following screen:<br>  |
| 3 | Modify the parameters you wish to change:<br>● enter the number of attempts you wish to be made for failure recovery in the **Number of tries** field,<br>● enter the timeout delay to be used for failure recovery in the **Timeout (ms)** field. |
| 4 | Confirm with **OK**.<br>**Note**: Any modification to the simulator communication parameters will only actually be taken into account if it is confirmed by clicking **OK** in the **Set Address** screen. Clicking **Cancel** cancels acknowledgment of any modifications. |

## Procedure for accessing bandwidths

You may divide up the bandwidth quota allocated to Unity Pro for the selected communication medium between the different functions available in online mode.

This screen is used to define the percentage of bandwidth allocated to each of the 4 categories of functionality in online mode:

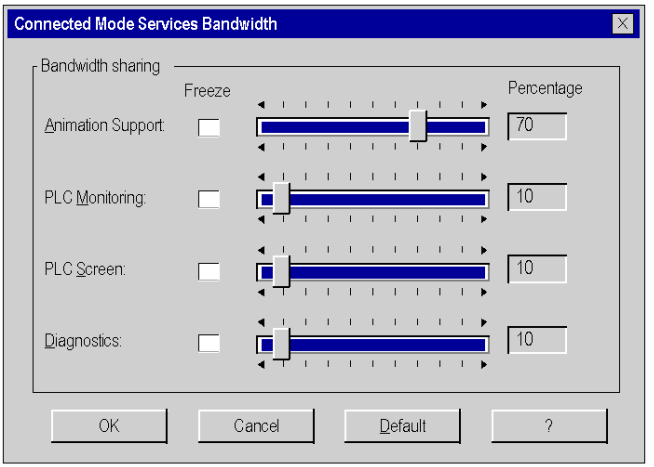
- **Animation Support**: animation tables, operator screens, animation of language editors, application-specific screens. The higher the percentage allocated, the higher the frequency with which the data in the PLC will be scanned.
- **PLC Monitoring**: monitoring of the overall operating mode of the PLC (Run or Stop), or of its tasks for the "Program debugging" function. The higher the percentage allocated, the higher the frequency with which the state of the PLC and its tasks will be scanned.

- **PLC Screen** (PLC debug screen): the higher the percentage allocated, the higher the frequency with which the PLC and application information will be scanned.
- **Diagnostics** (Diagnostic Viewer). The higher the percentage allocated, the higher the frequency with which the alarms in the PLC will be acquired.

This screen is used to optimize performance in online mode, by adjusting the parameters to suit:

- the characteristics of the project loaded onto the PLC: numerous and frequent diagnostics alarms, many animation screens open at once
- and the use made of online mode: the most frequently used functions.

Procedure:

| Step | Action |
|---|---|
| 1 | Access the **Set Address** screen *(see page 101)*. |
| 2 | Click on the **Bandwidth** button to display the following screen:<br>  |
| 3 | Using the sliders, adjust the parameter values to suit the characteristics of your project and the functions you use most frequently in online mode. |
| 4 | Click on the **Default** button if you want to revert to the default values. |
| 5 | The **Freeze** option is used for fine adjustment (in increments) of the parameters to suit the characteristics of the project loaded in the PLC.<br>It is used to retain the value of a parameter once the response time of the corresponding function has been determined, and then adjust only the parameters corresponding to the other functions. |
| 6 | Confirm with **OK**.<br>**Note**: Any modification will only actually be taken into account if it is confirmed by clicking **OK** in the **Set Address** screen. Clicking **Cancel** cancels acknowledgment of any modifications. |

# Project comparison

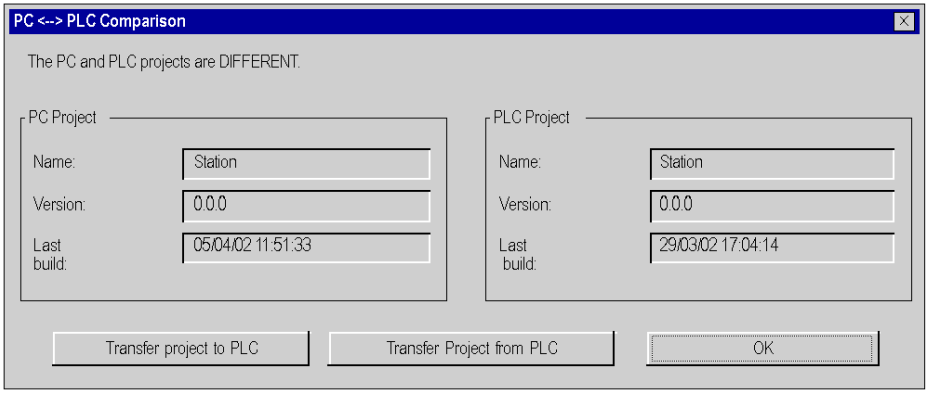
## At a Glance

Project comparison provides you with a concise overview of the possible differences between a project embedded in the PLC and a project in the terminal. For each project, the following information is displayed:

- **Name**,
- **Version**,
- **Last build**: date and time the executable was last generated.

## Procedure

You must perform the following actions to launch project comparison:

| Step | Action |
|---|---|
| 1 | Activate the **PLC** → **Compare** command, which displays the screen below.<br>  |
| 2 | Confirm with **OK**. |

## Different projects

If the projects are different, 2 commands let you carry out a transfer between the terminal and the PLC or vice versa, thus changing to Equal online mode:

- **Transfer Project to PLC**, transfer the project from the terminal to the PLC,
- **Transfer Project from PLC**, transfer the project from the PLC to the terminal.

**NOTE:** When the transfer is not possible, the corresponding button is grayed out (for example, the **Transfer Project from PLC** command is grayed out when the project embedded in the PLC is invalid or when information required for uploading is missing or invalid).

# Conversion of Legacy Applications/Projects

## Introduction

There are 2 possibilities to convert legacy applications to Unity Pro.

- Global conversion provides to convert a legacy application (PL7, Concept) to Unity Pro as a whole.
- Partial conversion with the conversion wizard provides to select parts of a legacy application to be converted and to control the conversion result.

To convert a legacy application to Unity Pro, you must at first export the application out of the legacy software (PL7, Concept).

**NOTE:** In case of global conversion with a default hardware configuration (see below) as well as in case of partial conversion, the hardware configuration must be edited and completed by the user.

## Global Conversion

To convert a legacy application as a whole you select the file exported out of the legacy software in Unity Pro via **File** → **Open**.

The respective converter (PL7 Converter, Concept Converter) automatically generates a Unity Pro application.

For detailed information on the converters, please refer to *Unity Pro Software -> PL7 Converter* or *Unity Pro Software -> Concept Converter*.

- Global conversion for Premium and Quantum contains also a hardware configuration.
- Applications with PLCs which do not have a direct equivalent in Unity Pro (Compact, Momentum, TSX Micro) should be converted using the partial conversion with the conversion wizard.

However, if global conversion is chosen, a default hardware configuration is generated.

- For global conversion of Compact applications you can select either Quantum or Modicon M340 as target PLC. Please refer to *Importing an LL984 Application* *(see page 117)*.
- Global conversion of Momentum applications contains only a default Quantum hardware configuration with a CPU (140 CPU 534 14A/U) and a power supply (140 CPS 424 00). If needed the default CPU can be replaced.
- Global conversion of TSX Micro applications contains only a default Modicon M340 hardware configuration with a CPU (BMX P34 2020) and a power supply (BMX CPS 2000). If needed the default CPU can be replaced.

## Partial Conversion

To convert a legacy application partially to Unity Pro, use the conversion wizard.

You can launch the conversion wizard in Unity Pro via **Tools** → **Convert Partially**.

With the conversion wizard you can:
- select the parts of the application to be converted
- remap I/O objects before conversion
- adapt concurrently the hardware configuration of the new application in Unity Pro
- modify the amount of used memory in the CPU

The partial conversion is possible for TSX Micro, Premium, Quantum, Momentum and Compact applications.

With partial conversion for Compact and Momentum applications it is recommended to prepare a Modicon M340 hardware configuration.

For detailed information, please refer to *Unity Pro Software -> Operating Modes -> Conversion Wizard, page 1777*.

# Importing an LL984 Application

## Introduction

When you import an application to Unity Pro from other programming software, some functions may behave differently.

For example:
- Cold start behavior of Unity Pro LL984 applications is different from legacy LL984 applications.
- Imported Concept IEC coils behave differently than imported LL984 coils.
- Unit Pro MATH functions might not behave the same as legacy LL984 (overflow, underflow, etc.).

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

- After you imported an application program, edit the result carefully to check that all the logic elements have been ported completely and are located correctly in the program.
- Test the application before running it to check that it performs as expected.
- If necessary, edit the program to add logic elements that are missing, remove code that is not necessary, and replace logic elements that behave differently from their intended purpose.

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

## General Message after Import

After the import check the log file *(see page 124)* for error and warning messages.

You have to **Analyze** and **Build** the project.

## Import

To import an LL984 application use the **File** → **Open** menu command.

Please refer to **Open** dialog *(see page 85)*.

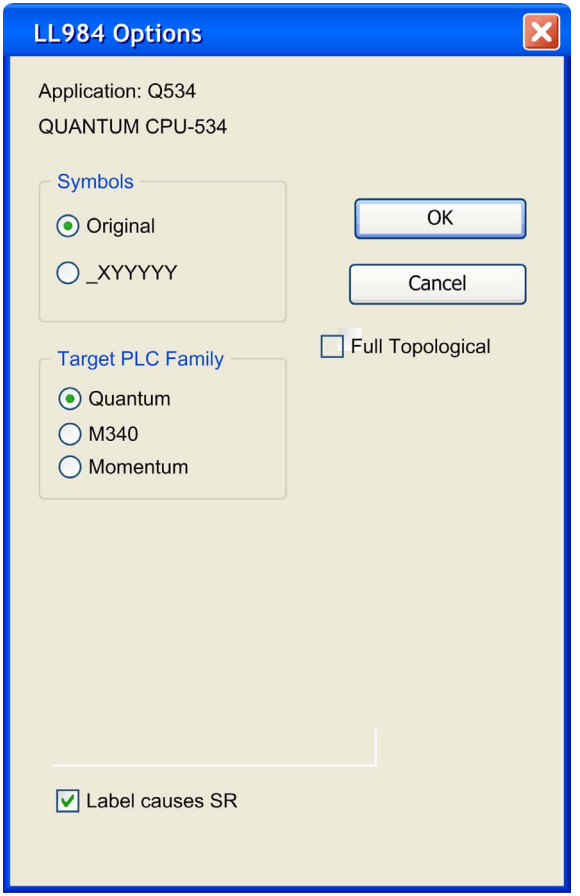
You can import the following legacy LL984 applications:
- Concept Projects (**.ASC*)
- Modsoft Projects (**.CFG*)
- ProWORX32 Databases (**.PWX*)
- ProWORX Nxt (**.DCF*)

**NOTE:** Imported MSTR functions are not designed to work with the Modicon M340 CPUs. After an application is converted to Unity Pro, it must be manually updated. The MSTR function must be replaced by the READ_VAR and WRITE_VAR functions.

For further information please refer to Import / Export of LL984 Applications *(see LL984 Editor, Reference Manual, LL984 Specifics)*.

## LL984 Options Dialog (-> Target PLC Quantum)

After selecting a legacy LL984 application file, the following **LL984 Options** dialog is displayed. The following graphic shows an example with a legacy Quantum LL984 source application and a Quantum selected as target PLC family.




| Element | Description |
|---|---|
| **Application** | Displays the name of the legacy application to be imported. |
| **Source PLC Family** | Displays the PLC family of the legacy application to be imported. |
| **OK** | Starts the import. |
| **Cancel** | Displays a message `...“Convert” step failed`, no application is imported, and the following message is generated in the import log file: `“Conversion terminated by User”`. |

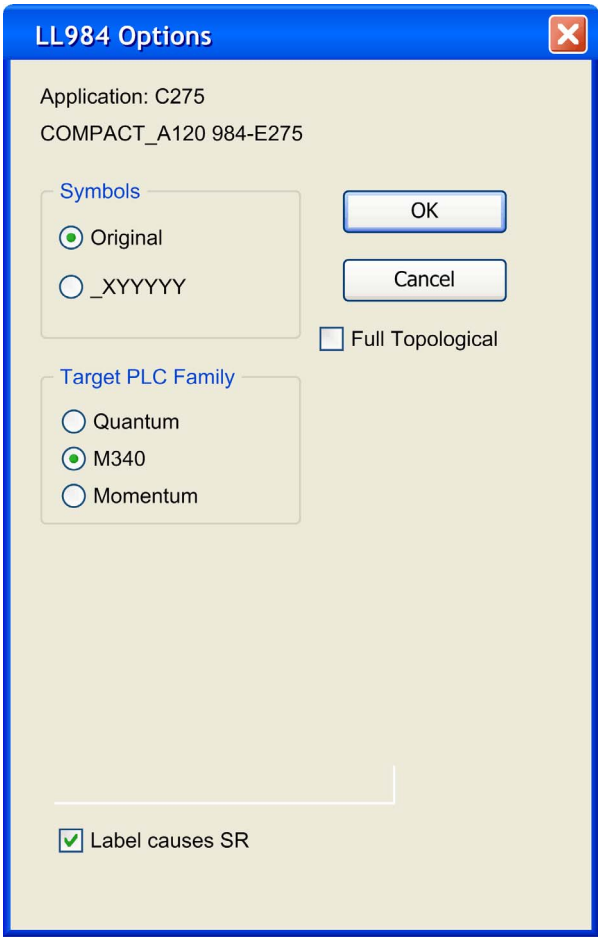
| Element | Description |
|---|---|
| **Full Topological** | This check box is only used to convert applications to Modicon M340 PLCs. |
| **Symbols** | |
| **Original** | Select this option to use the original variable names of the legacy application in your new Unity Pro application. |
| **_XYYYYY** | Select this option to automatically convert the legacy variable names.<br>The new variable names will be composed according to the following list:<br>● **_**: an underscore will be the leading sign of the converted variables.<br>● **X**: represents the data type in the memory area.<br>❍ **0**: %M (memory bit)<br>❍ **1**: %I (discrete input)<br>❍ **3**: %IM (input word)<br>❍ **4**: %MW (memory word)<br>● **YYYYY**: represents the address number.<br>For example **_400001** stands for%MW1. |
| **Target PLC Family** | |
| **Quantum** | Select this option to use the Quantum as target PLC. |
| **M340** | Select this option to use the Modicon M340 as target PLC.<br>The Modicon M340-specific parameters for full topological addressing are listed in a separate table below. |
| **Momentum** | Select this option to use the the Modicon Momentum under Unity Pro as target PLC. |
| **Label causes SR** | Select this option to import a segment containing labels as an SR segment, even if its scheduling does not express this.<br>Labels are allowed only in SR segments.<br>For details please refer to Calling Subroutines *(see page 1175)*. |

## LL984 Options Dialog (-> Target PLC Modicon M340)

Converting a legacy LL984 application to a Modicon M340 application you can determine the kind of addressing. By default Mixed topological / State RAM addressing is used (option **Full Topological** not selected).

**Mixed Topological / State RAM Addressing (-> Modicon M340)**

The following graphic shows an example for converting a legacy Compact application to a Modicon M340 application with Mixed topological / State RAM addressing (option **Full Topological** not selected).



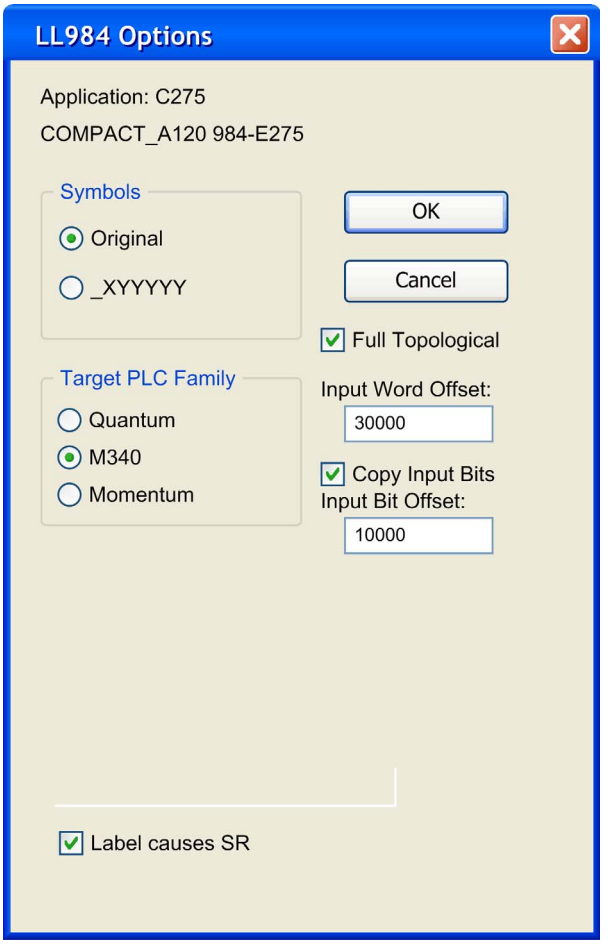

**NOTE:** If you want to import a legacy LL984 Compact application which uses Modbus request to communicate with an HMI, you have to use State RAM addressing to preserve the Modbus exchange between PLC and HMI.

**Full Topological Addressing (-> Modicon M340)**

To convert a legacy LL984 application to a Modicon M340 application with full topological addressing, the option **Full Topological** has to be selected.

- All word variables get a declaration with a symbol and the topological address in the data editor, if not already present.
- **Input Words** are copied to a mirror area of memory words, allowing SCADA access.
- Discrete words are directly addressed with topological addresses in the programs, if no symbols are defined for them, showing in this way directly their hardware connection to the controller.
- If SCADA access is required to discrete inputs, the input bits can be copied as well to a mirror area of memory bits by checking **Copy Input Bits**.
- SCADA access for inputs with **Full Topological** checked needs modified addresses in the SCADA system – the offsets in the dialog must be added to the original offsets and the memory areas must be used instead of the input areas.

The following graphic shows an example for converting a legacy Compact application to a Modicon M340 application with full topological addressing.



For converting a legacy LL984 application to a Modicon M340 application with full topological addressing, the following additional parameters are used:

| Element | Description |
|---|---|
| **Full Topological** | Activate this option to convert input word addresses (3x) of the legacy LL984 application to a separate, mirrored area of the %MW areas of the target Modicon M340 application.<br>If this option is not activated, the legacy LL984 application is converted into a State RAM Modicon M340 application. |
| **Input Word Offset** | Enter the offset for the input word addresses (3x) in the mirrored area of the %MW area of the target Modicon M340 application. |
| **Copy Input Bits** | Activate this option to additionally convert input bit addresses (1x) to the mirrored area of the %M area of the target Modicon M340 application. |
| **Input Bit Offset** | Enter the offset for the input bit addresses (1x) in the mirrored area of the %M area of the target Modicon M340 application. |

**NOTE:** With this **Full Topological** option activated you must manually change the access to the %I and %IW addresses in the SCADA components.

### Converting a Legacy Compact Application to Modicon M340 with Full Topological Addressing

During conversion of a legacy Compact application to a Modicon M340 application with full topological addressing, the import automatically creates separate ST code sections for input and output operations with copying commands to transfer I/O data to the mirrored memory areas.

Please refer to *Module Emulation (see LL984 Editor, Reference Manual, LL984 Specifics)*.

### Import Behavior

During import of a legacy LL984 application, problems may be detected:

- controller family not supported
- I/O family not supported
- I/O module not supported
- rack count exceeded
- instruction/function block not supported
- non-coil in column 11

Generally such an issue does not stop the import.

Instead a message is displayed for each of these detected problems in the **Import/Export** tab of the **Output Window** at the bottom of the Unity Pro user interface.

Double-click such a message in the **Output Window** to access the respective dialog in your application.

If detected problems occur the following is done during import:

- Not supported controllers or I/O families are replaced by a default Quantum/Modicon M340 configuration.
- Not supported I/O modules are ignored in case of Quantum as target PLC family.
- Exceeding racks and their modules are ignored.
- Not supported instructions are represented by empty DFBs that look identical to the LL984 instruction. In such a case proceed as described under *Not Supported Legacy LL984 Instructions and User Loadables (see LL984 Editor, Reference Manual, LL984 Specifics)*.
- Equation networks with questionable conversions (for example implicit type cast) will contain the imported equation statement and an additional message.
  Equation networks which are not compliant with Modicon M340 variable alignment rules, will cause a build error during **Analyze**.
  For details please refer to Equation Networks *(see LL984 Editor, Reference Manual, LL984 Specifics)*.
- The import of a legacy application that contains duplicated identifiers is aborted. For details please refer to Duplicated Identifiers *(see LL984 Editor, Reference Manual, LL984 Specifics)*.

If parts of a legacy application are not imported, after **Analyze** and **Build** of the project an appropriate message is displayed. In this case you have to adapt the application manually. (For example in case of Modicon M340 by selecting **Mixed topological and State RAM** in the *Memory Tab (see page 1219)* of the modules and subsequently fixing the related variables).

## Import Log File

After import a message informs you that the conversion and import is finished and reminds you to have a look at the import log file.

The import log file is stored in the root directory of your application.

You have to **Analyze** and **Build** the project.

# Transferring the project between the terminal and the PLC

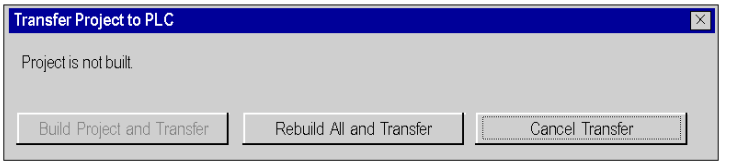
## At a Glance

Transferring a project enables you to:

- copy the current project from the terminal to the memory of the current PLC (PLC whose address is selected)
- copy the embedded project from the current PLC to the terminal. It then becomes the current Unity Pro project

## Transfer from the terminal to the PLC

Perform these actions to transfer the current project to a PLC:

| Step | Action |
|---|---|
| 1 | Use the **PLC → Set address** command to define the current PLC to which you would like to transfer your project *(see page 101)*. |
| 2 | Change to online mode *(see page 99)*. |
| 3 | Activate the **PLC → Transfer Project to PLC** command, which displays the corresponding screen *(see page 126)*. |
| 4 | Check the **PLC Run after Transfer** box if you want the PLC to switch automatically to Run after the transfer. |
| 5 | Activate the **Transfer** command. |
| 6 | If the project has not been generated in advance, the screen below allows you to generate it before the transfer (**Rebuild All then Transfer**) or interrupt the transfer (**Cancel Transfer**).<br>  |
| 7 | The transfer progress is displayed on screen. At any moment, you can interrupt the transfer by using the **Esc** key, which invalidates the PLC project.<br>Note: In the event that the project is transferred to a Flash Eprom memory card, the transfer can take several minutes. |

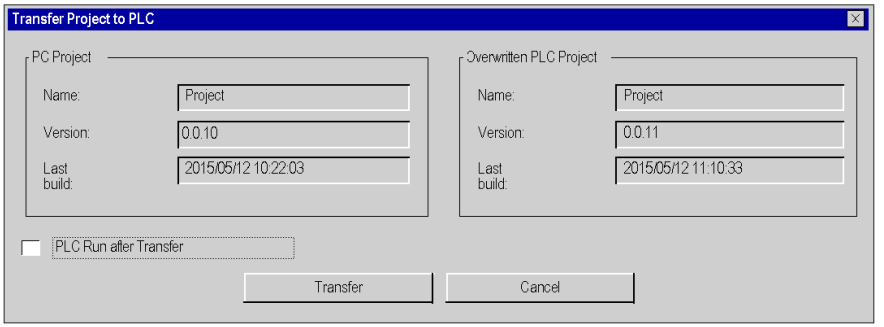
## Transfer from the PLC to the terminal

Perform these actions to transfer the contents of the PLC to the terminal:

| Step | Action |
|---|---|
| 1 | Use the **PLC → Set address** command to define the current PLC to which you would like to transfer your project *(see page 101)*. |
| 2 | Change to online mode *(see page 99)*. |
| 3 | Activate the **PLC → Transfer Project from PLC** command to display the corresponding screen *(see page 127)*. |
| 4 | The transfer progress is displayed on screen. At any moment, you can interrupt the transfer by using the **Esc** key. |

## Terminal to PLC transfer screen

The screen used to transfer the project from the terminal to the PLC is shown below:



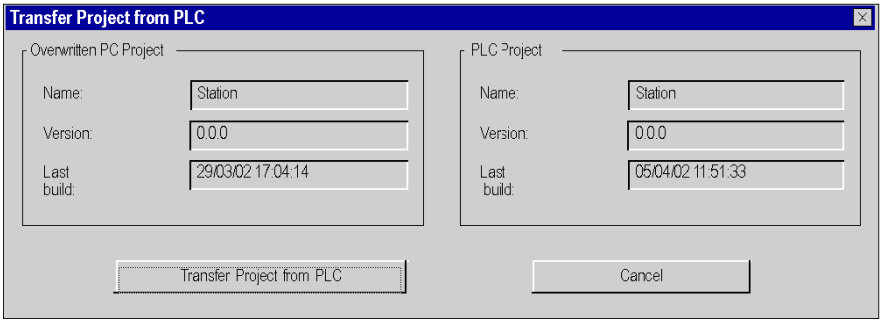
## Description of the parameters

The following table describes the parameters of the Terminal to PLC transfer screen.

| Parameter | Description |
|---|---|
| **PC Project** | Identification field for the project in the terminal: name, version, date and time the executable was last generated. |
| **Overwritten PLC Project** | Identification field for the project embedded in the PLC: name, version, date and time the executable was last generated. This project will be overwritten or updated by the transfer. |
| **Compatibility** | Restriction of compatibility (only for ascending compatibility). The configuration declared in the current project is slightly different from the real PLC configuration, but remains compatible. |
| **PLC Run after Transfer** | If this box is checked, the PLC changes to Run after the transfer. |
| **Transfer** | Transfer command. |
| **Cancel** | Command used to cancel the transfer. |

## PLC to Terminal transfer screen

The screen used to transfer the project from the PLC to the terminal is shown below:



## Description of the parameters

The following table describes the parameters of the PLC to terminal transfer screen.

| Parameter | Description |
|---|---|
| **Overwritten PC Project** | Identification field for the project in the terminal: name, version, date and time the executable was last generated. This project will be overwritten or updated by the transfer. |
| **PLC Project** | Identification field for the project embedded in the PLC: name, version, date and time the executable was last generated. |
| **Transfer Project from PLC** | Transfer command: the project is transferred. |
| **Cancel** | Command used to cancel the transfer. |

## Incompatibility during a transfer

During transfer from the terminal to the PLC, incompatibility between the current project and the PLC configuration triggers display of a message.

During transfer from the PLC to the terminal, incompatibility between the project embedded in the PLC and the version of Unity Pro also triggers display of a message.

# Update of Upload Information

## Upload Information

Upload information consists of:

- Project upload information: graphic code for languages with LD contacts and chart of FBD function blocks, symbols of located and unlocated variables,
- Comments: variables, code sections, project, DFBs, etc.,
- Animation tables.

These are always saved when a project backup is performed on the terminal (STU or STA file), but may or may not be included with the executable code *(see page 519)* when building the project which is then transferred to the PLC.

## Use of Upload Information

Upload information should only be loaded in the PLC where this is strictly necessary, in order to optimize the available memory size and save this for the executable code, and also to improve the performance of modifications in online mode.

**In the coding and debug phase**, this information is generally present on the workstation and therefore does not need to be transferred to the PLC, thus achieving a real gain in performance.

**In the maintenance and operation phase**, this information should be included in the PLC where the application needs to be restored on empty terminals (i.e. that do not have the project STU or STA file). However, in addition to the gain in memory space, the absence of the Upload information also forms a protection against reading and writing.

## Choice of Upload Information

By default, the upload information necessary to rebuild the project program is not embedded in the PLC.

You can choose the information to have embedded:

- The information necessary for uploading the project,
- Comments (variables and types),
- The animation tables.

To access the Upload information *(see page 519)* select: **Tools → Project Settings → PLC Embedded data**

## Update

When making modifications in online mode, it is possible to choose between 2 modes of updating the Upload information:

- Automatic update; in which case the update of Upload information is implicit on each build,
- Update on user request; in which case the update is performed explicitly, by the clicking **PLC → Update Upload Information**.

The way in which upload information is updated is chosen in the Project Settings screen *(see page 557)* and is accessible by: **Tools → Project Settings → PLC Embedded data**

**NOTE:** The upload information is updated implicitly whenever a terminal project is transferred to the PLC (where this is included with the executable code).
On connection, a message asks you to confirm the update of the Upload information, if this has not already been performed.

# Sending a command to the PLC

## At a Glance

The **Run/Stop** and **Init** commands enable the project to be controlled from the terminal when it is stored on a target PLC:

- **Run/Stop** launches or stops execution of the project, *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- **Init** initializes the project.

## Procedure

You must perform the following actions to control the PLC:

| Step | Action |
|---|---|
| 1 | Use the **PLC → Set address** command to define the target PLC that you wish to control *(see page 101)*. |
| 2 | Change to online mode. |
| 3 | Activate the **PLC → Run/Stop** command to launch or stop execution of the project or the **PLC → Init** command to initialize the project. |
| 4 | Confirm the command. |

# Project Backup Management for Modicon M340

## At a Glance

Automatic and manual accesses are carried out between the Schneider BMX RMS ••••• memory card and Unity Pro or the PLC to:

- restore a project in the PLC
- save the PLC project on the memory card
- compare the PLC project and the memory card project

| ***NOTICE*** |
| --- |
| **INOPERABLE MEMORY CARD**<br>Do not format the memory card with a non-Schneider tool. The memory card needs a structure to contain program and data. Formatting with another tool destroys this structure.<br>**Failure to follow these instructions can result in equipment damage.** |

## Restore Function

The restore function is carried out as follows:

- Automatically:
  - after a power cycle
  - on a warm start
  - on a cold start
- Manually, using the Unity Pro command **PLC → Project Backup... → Backup Restore**.

**NOTE:** When you insert the memory card in run or stop mode, power cycle to restore the project on the PLC.

| **⚠ WARNING** |
| --- |
| **UNEXPECTED BEHAVIOUR OF APPLICATION**<br>Ensure that the right user application is contained in the memory card before inserting it into the PLC.<br>If the application contained in the memory card includes the RUN AUTO option, then the PLC will automatically restart in RUN mode with this application as soon as the inserted memory card content is restored to the PLC.<br>**Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

## Save Function

The save function is carried out as follows:

- Automatically, after:
  - a download, if the memory card is present and not write-protected
  - online modification, if the memory card is present and not write-protected
  - detection of a system bit %S66 rising edge
- Manually, using the command **PLC → Project Backup... → Backup Save**.

**NOTE:** If you remove the memory card when the backup is in progress, the program on the memory card is lost. With system bit %S65 you can disable processor access to the memory card. If the memory card is not present or write-protected, the download in Processor memory is allowed but the online modification is not allowed.

## Compare Function

You can compare your project with the one contained on the memory card. To do so, use the command **PLC → Project Backup... → Compare**.

## Memory Card State and Diagnostic

Unity Pro and the PLC front panel display the status of the memory card:

| Display on | Icon/LED State | Description |
|---|---|---|
| Status bar at bottom of Unity Pro |  | Project on memory card is different from PLC project. |
| |  | Memory card is not present on the PLC or is inoperable. |
| | no icon | Project in memory card is equivalent to PLC project. |
| LED on bottom of memory card slot | On | Access to the memory card is enabled. |
| | Flashing | Activity between the memory card and the PLC. |
| | Off | Access to the memory card is disabled. You can remove the memory card. |
| CARDERR LED on PLC front panel | On | Memory card is missing or memory card is not detected, or project on memory card is different from PLC project. |
| | Off | Memory card is detected, and project in memory card is equivalent to PLC project. |

# Project backup for Premium

## At a Glance

Certain PLCs Premium allow you to save the project program contained in the executable zone *(see Unity Pro, Program Languages and Structure, Reference Manual )* (RAM) of the PLC in the backup memory zone.

The available backup memory card is the **TSX MFP B 096K**.

Unity pro enables the use of these functions:

- Comparison
- Automatic restore (shaded field) when the PLC powers up, if the project program is different
- Save PLC program on the memory card
- Clearing of card data

**NOTE:** The backup function is not available when the project is already being executed on a RAM or EPROM memory card.

## Comparison

You can compare your project with the one contained in the backup memory. To do this, perform the following actions:

| Step | Action |
|---|---|
| 1 | Activate the command **PLC → Project backup → Compare...**.<br>**Result**: A dialog box appears informing you of the result of the comparison. |

## Backup Memory Card → RAM Transfer

You Must Perform the Following Actions to Restore the Contents of the Backup Memory to the Executable Zone of the PLC:

| Step | Action |
|---|---|
| 1 | Set the **WP** switch on the backup memory card to **ON** (lock). |
| 2 | Insert the backup memory card into the PLC. |
| 3 | **Result**: The memory card automatically transfers the contents of the backup memory to the executable zone of the PLC. |

**NOTE:** Inserting the memory card in its slot triggers a cold start of the PLC.

### RAM → backup memory card transfer

You must perform the following actions to transfer the project program to the backup memory:

| Step | Action |
|---|---|
| 1 | Set the **WP** switch on the backup memory card to **OFF** (unlock). |
| 2 | Insert the backup memory card into the PLC. |
| 3 | Activate the command **PLC → Project backup → Save...**.<br>**Result**: A command confirmation screen is displayed. |
| 4 | Confirm with **OK**. |
| 5 | Remove the backup memory card from the PLC. |
| 6 | Set the **WP** switch on the backup memory card to **ON** (lock). |

### Clearing procedure

You must perform the following actions to clear the backup memory:

| Step | Action |
|---|---|
| 1 | Set the **WP** switch on the backup memory card to **OFF** (unlock). |
| 2 | Insert the backup memory card into the PLC. |
| 3 | Activate the command **PLC → Project backup → Clear...**.<br>**Result**: A command confirmation screen is displayed. |
| 4 | Confirm with **OK**. |

# Save/Restore Data Between a File and the PLC

## At a Glance

The data transfer function allows you to store the value of the following data *(see Unity Pro, Program Languages and Structure, Reference Manual )* in a file:

- boolean-type located data: `%Mi`
- word-type located data: `%MWi`
- unlocated variables
- function block instances

There are 2 types of data files:

- DAT:
  - Saving PLC data to a *.DAT file is allowed for all previous data types. *.DAT contains only data blocks.
  - Restoring all data blocks from a *.DAT file to the PLC is allowed if the project has not been built since the Save was done.If a build was done, only BOOL and WORD located data (%Mi and %MWi) are restored.
- DTX (requires Unity Pro 6.0 of higher):
  - Saving PLC data to a *.DTX file is allowed for all previous data types. In addition, application references (at the time of the Save) are saved.
    **NOTE:** I/O and device DTT objects cannot be saved in DTX data files.
  - Restoring data from a *.DTX file is allowed even if the application has been built and/or data has been modified since the Save was done. Compatibility Rules for Restore Using a *.DTX File *(see page 140)* explains the restoration process and the types of data that cannot be restored.

**NOTE:** It is recommended to convert any *.DAT files to *.DTX file by restoring the.DAT file and then saving using the *.DTX format.

**⚠ CAUTION**

**LOSS OF DATA**

Before transferring unlocated variables and function block instances data to the PLC, ensure that your application is in a compatible state with the saved data.

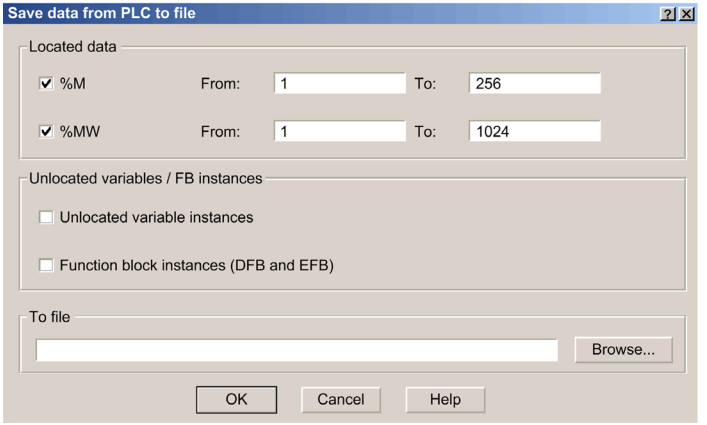
**Failure to follow these instructions can result in injury or equipment damage.**

## Procedure for Saving PLC Data

If the PLC is in RUN mode, the saving of data from the PLC to a file may need several application cycles and the data, in the file, may be de-synchronized, which may affect application execution during restoration.

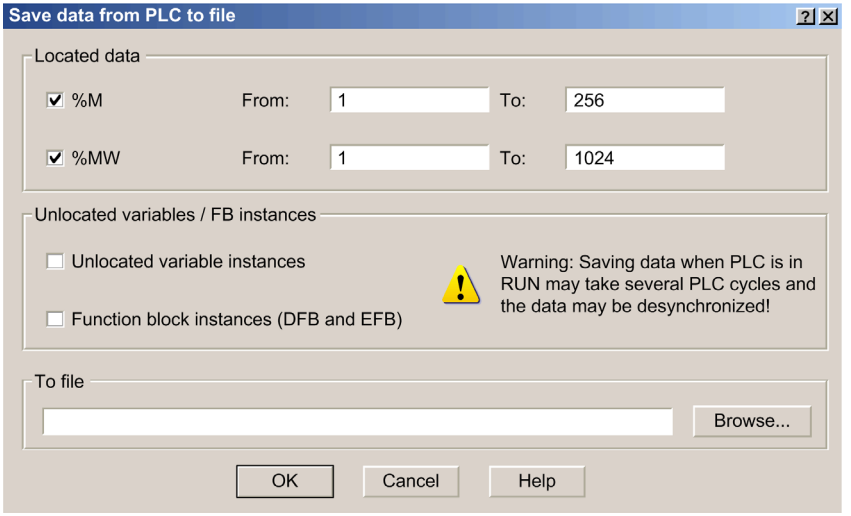
| ⚠ CAUTION |
| --- |
| **UNEXPECTED BEHAVIOUR OF APPLICATION**<br>Before saving the data, verify the impact of the save on the application execution.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

You must perform the following actions to save the data from the PLC to a file:

| Step | Action |
| --- | --- |
| 1 | To enable the menu item **Save Data from PLC to File**, the following must be true:<br>● The PLC is loaded with an application.<br>● The PLC is in RUN or STOP mode.<br>● Unity Pro is running and is connected to the PLC.<br>● The Application built in Unity Pro is equal to those in the PLC.<br>● The Application in Unity Pro is in the built state. |
| 2 | Activate the **PLC → Save Data from PLC to File** command.<br>**Result**: The **Save Data from PLC to File** dialog (CPU in STOP mode) opens.<br>  <br>For more information, refer to the corresponding screen *(see page 138)*. |
| 3 | Using the checkboxes, select the data type(s) to be transferred. For located data you must also define the range of data to be transferred by entering the start and end address of the range. |
| 4 | Define the path and the name of the file to which the data is to be saved. |
| 5 | Confirm with **OK**. |

## Data Transfer Dialog for Transfers to a File

The **Save Data from PLC to File** dialog (CPU in RUN mode) is shown below:




The following table shows the parameters of the **Save Data from PLC to File** dialog.

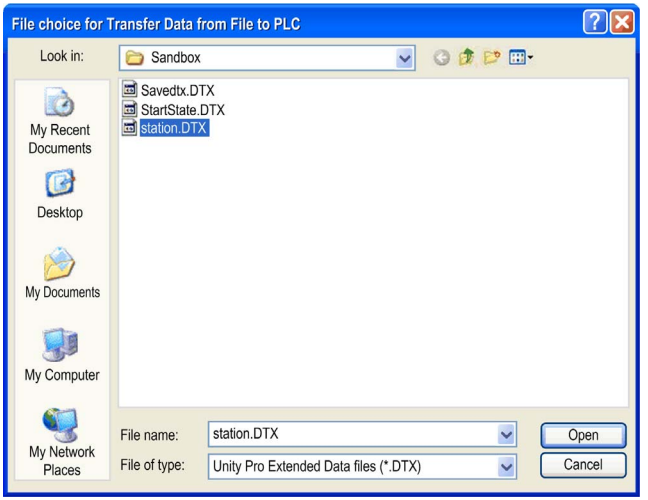
| Parameter | Description |
|---|---|
| Content | This field is used to define the type and range of the data to be transferred.<br>The checkboxes are used to define the type of located data. **From**: and **To**: fields are used to define the range for located data.<br>Default values:<br>● **%M** and **%MW** checkboxes are checked<br>● **%M** and **%MW** range is set to the PLC values<br>● **Unlocated variables instances** and **Function block instances (DFB and EFB)** are not checked |
| To file | This field is used to define the path and name of the file. |
| Browse | This button can be used to browse the disk to define the file name and path. |

## Procedure for Restoring Data to PLC

If the PLC is in RUN mode, the restoring of data to the PLC from a file may need several application cycles and the data may be de-synchronized, which may affect application execution.

| ⚠ CAUTION |
| --- |
| **UNEXPECTED BEHAVIOUR OF APPLICATION**<br>Before restoring the data, verify the impact of the store on the application execution.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

You must perform the following actions to restore the data from a file to the PLC:

| Step | Action |
| --- | --- |
| 1 | To enable the menu item **Restore Data from File to PLC**, the following must be true:<br>● The PLC is loaded with an application.<br>● The PLC is in RUN or STOP status.<br>● Unity Pro is running and is connected to the PLC.<br>● The Application built in Unity Pro may be different than those in the PLC.<br>● The Application in Unity Pro is built state. |
| 2 | Activate the **PLC → Restore Data from File to PLC** command.<br>**Result**: The **Restore Data from File to PLC** dialog opens.<br>  |
| 3 | Select the file to be restored to the PLC. |
| 4 | Confirm with **OK**. |

# Compatibility Rules for Restore Using a *.DTX File

## At a Glance

The DTX file allows a restore even if there have been changes in the application since the save was done.

When a restore is performed, if the build number of the application is greater than the version of the DTX file, Unity Pro analyses the content of the file to verify the compatibility with variables of the PLC application.

The following tables presents the rules followed for restoring variables when there is a difference between the *.DTX file and the PLC application. This condition results from building the application after the data was saved in the *.DTX file.

| ⚠ CAUTION |
| --- |
| **UNEXPECTED BEHAVIOUR OF APPLICATION**<br>Before restoring the data, verify the impact of the store on the application execution.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

## PLC versions compatibility (Premium, Quantum , M340, M580)

As memory block allocation is different on PLCs version <= 2.0 or > 2.0:

- If the DTX file has been created from a PLC version 2.0 or < 2.0, data can't be restored on PLC version 2.1 or later.
- If the DTX file has been created from a PLC version 2.1 or later , data can't be restored on PLC version 2.0 or < 2.0.

### Simple Variable Compatibilities

The following table presents the rules followed for restoring simple variables:

| Case | Data File Contents | PLC Application Contents | Restore Behavior | Unity Pro Message |
|---|---|---|---|---|
| 1 | Var1 | Deleted | Var1 is not restored | Var1: this variable no longer exists. |
| 2 | - | New Var2 | Var2 is kept in memory | - |
| 3 | Var3 | Var3 was renamed into Var3-3 (name changed) | The restore CANNOT BE PERFORMED due to technical limitations.<br>Var3 is deleted and Var3-3 is a new variable. | Var3: this variable no longer exists. |
| 4 | Var4 is a simple compatible type. | Var4 was changed to a different simple type. Source value is compatible with target type. | The Var4 is converted into the new type, which has the same value as source. | Var4: type of the variable is changed (from TYPE1 to TYPE2). Types are compatible. |
| 5 | Var5 is a simple compatible type. | Var5 was changed to a different simple type. Source value is not compatible with target type. | The restore CANNOT BE PERFORMED.<br>Var5 is converted into a different type (the target) and its value is either:<br>● longer than target type<br>● not compatible with target type | Var5: type of the variable is changed (from TYPE1 to TYPE2) but value may not be compatible.<br>+<br>Var5: value not compatible for conversion between TYPE1 and TYPE2. |
| 6 | Var6 is a simple incompatible type. | Var6 was changed into another type. | Restore of original value CANNOT BE PERFORMED.<br>Var6 retains its application value. | Var6: type of the variable is changed (from TYPE1 to TYPE2). But types are not compatible. |

## Structured Variable Compatibilities

The following table presents the rules followed for restoring structured variables:

| Case | Data File Contents | PLC Application Contents | Restore Behavior | Unity Pro Message |
|---|---|---|---|---|
| **DDT Types** | | | | |
| 7 | Var7 is a DDT type. | Var7 was changed to a simple type. | The restore CANNOT BE PERFORMED because type is not compatible. | Var7: type of the variable is changed (from TYPE1 to TYPE2). But types are not compatible. |
| 8 | Var8 is a simple type. | Var8 was changed to a DDT type. | The restore CANNOT BE PERFORMED because type is not compatible. | Var8: type of the variable is changed (from TYPE1 to TYPE2). But types are not compatible. |
| 9 | Var9 is a DDT type. | Var9 is changed to another DDT type. | The restore CANNOT BE PERFORMED because type is not compatible. | Var9: type of the variable is changed (from TYPE1 to TYPE2). But types are not compatible. |
| 10 | Var10 is a DDT type. | The DDT type has changed. | Make the mapping of the field (see the cases from 1 to 9 and cases from 11 to 16 for array type sub-fields). | Var10.SubField1: … (see cases 1 to 9 and 11 to 16). |
| **ARRAY Types** | | | | |
| 11 | Var11 is an array. | Var11 is an array of the same type, same starting index but longer. | Transfer the data file values. The rest of the longer application array is kept. | - |
| 12 | Var12 is an array. | Var12 is an array of the same type, same starting index but shorter. | Transfer the data file values until target array is full. The rest of the source array is lost. | Var12: length of ARRAY… array (range # 2) is shorter. Last indexes will be lost. |
| 13 | Var13 is an array. | Var13 is an array of the same type, but starting index is lower or a different type. Ending index is the same | The restore CANNOT BE PERFORMED because starting indexes must be identical. | Var13: low bound of ARRAY… array (range #2) has changed. Cannot restore. |
| 14 | Var14 is an array. | Var14 is an array of the same type, but starting index is higher or a different type. Ending index is the same | The restore CANNOT BE PERFORMED because starting indexes must be identical. | Var14: high bound of ARRAY… array (range #2) has changed. Cannot restore. |

| Case | Data File Contents | PLC Application Contents | Restore Behavior | Unity Pro Message |
|---|---|---|---|---|
| 15 | Var15 is an array. | Var15 is an array of the same type, but the lower and upper indexes are shifted. The length is the same. | The restore CANNOT BE PERFORMED because starting indexes must be identical. | Var15: bounds of ARRAY… array (range #2) have changed. Cannot restore. |
| 16 | Var16 is an array. | The type of the array has changed | See cases 1 to 10 and 7 to 10. | See cases 4 to 10 and 11 to 16. |

## Elementary Type Compatibilities

BOOL, BYTE, INT, UINT, DINT, UDINT, WORD and DWORD are compatible types if no lost of value are detected.

Examples are:

- a DINT variable can be changed into an INT variable if only the 2 first bytes of the DINT are used
- an INT variable can be changed into an INT variable if the value is compatible.

A BOOL to EBOOL change is OK because the types are compatible, but for an EBOOL to BOOL change the types are not compatible: the historic and forcing state are lost.

REAL, STRING, TOD, TIME, DATE and DT are not compatible types.

The table below presents the compatibility between the elementary types:

| | BOOL | EBOOL | BYTE | INT | UINT | WORD | DINT | UDINT | DWORD | REAL | STRING | TOD | TIME | DATE | DT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOOL | | YES | YES | YES | YES | YES | YES | YES | YES | NO | NO | NO | NO | NO | NO |
| EBOOL | PDL | | YES | YES | YES | YES | YES | YES | YES | NO | NO | NO | NO | NO | NO |
| BYTE | PDL | PDL | | YES | YES | YES | YES | YES | YES | NO | NO | NO | NO | NO | NO |
| INT | PDL | PDL | PDL | | PDL | YES | YES | YES | YES | NO | NO | NO | NO | NO | NO |
| UINT | PDL | PDL | PDL | PDL | | YES | YES | YES | YES | NO | NO | NO | NO | NO | NO |
| WORD | PDL | PDL | PDL | PDL | YES | | YES | YES | YES | NO | NO | NO | NO | NO | NO |
| DINT | PDL | PDL | PDL | PDL | PDL | YES | | PDL | YES | NO | NO | NO | NO | NO | NO |
| UDINT | PDL | PDL | PDL | PDL | PDL | PDL | PDL | | YES | NO | NO | NO | NO | NO | NO |
| DWORD | PDL | PDL | PDL | PDL | PDL | PDL | YES | YES | | NO | NO | NO | NO | NO | NO |
| REAL | NO | NO | NO | NO | NO | NO | NO | NO | NO | | NO | NO | NO | NO | NO |
| STRING | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | | NO | NO | NO | NO |
| TOD | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | | NO | NO | NO |
| TIME | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | | NO | NO |
| DATE | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | | NO |
| DT | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | NO | |

**NOTE: PDL** means Possible Data Loss.

## Compatibility of Aliases

The table below presents the rules followed for restoring aliases:

| Case | Data File Contents | PLC Application Contents | Restore Behavior | Unity Pro Message |
|---|---|---|---|---|
| 1 | Map1 is unlocated. %MW100 is used. | Map1 is located on %MW100. | Value of the data file for Map1 is set to %MW100. | Map1: restored at %MW100 |
| 2 | Map2 is located on %MW100. | Map2 is no longer located on %MW100, %MW100 is used. | Value of data file for Map2 is set at the new address of Map2 (unlocated data area). %MW100 of the application keeps its current value. | Map2: restored into unlocated data area. %MW100: keeps current value. |
| 3 | Map3 is located on %MW100. %MW200 is used | Map3 is located on %MW200. %MW100 is used. | Restores the value of %MW100 at address %MW 200. %MW100 of the application keeps its value. | Map2: restored from %MW100 to %MW200. %MW100: keeps current value. |
| 4 | Map4 is located on %MW100, but %MW area not exported. | Map4 is no longer located on %MW100, %MW100 is used. | Restore CANNOT BE PERFORMED because the source value is not in data file. | Map4: value at address %MW100 is not available. |
| 5 | Map5 is not located but in a unlocated area not exported. | Map5 is located on %MW100. | Restore CANNOT BE PERFORMED because the source value is not in data file. | Map5: value at address %MW100 is not available. |
| 6 | Map6 is located at %MW100 and %MW200 is used, but %MW area is not exported. | Map6 is now located on %MW200 and %MW100 is used. | Restore CANNOT BE PERFORMED because the source value is not in data file. | Map6: value at address %MW100 is not available. |
| 7 | Map7 does not exist or is an alias. | Map7 is located at %MW100. | Restore CANNOT BE PERFORMED to avoid the lost of the current variable. | Map7: mapped variable that didn't exist at the file generation or was an alias. %MW100 value not restored. |
| 8 | Topological addresses, %CH, %SW, etc. | - | - | MapVar: variable mapped on an unauthorized address (%CH0.2.0). |

## Restoration of FB Instances and Other Variables

The table below presents the rules followed for restoring certain special variables:

| Type of Entity | Unity Pro Message |
|---|---|
| FB Instances: for all fields (input, output, I/O, private, public) that is not a pointer, the fields are restored or not by using the rules above. If any field cannot be restored, the FB Instance is not restored | - |
| SFC variables cannot be restored. | Step1: SFC types (SFCSTEP_STATE) are not allowed for restore<br>Step1: SFC types (SFCCHART_STATE) are not allowed for restore |
| Hidden variables cannot be restored. | - |
| IODDT variables cannot be restored. | IOVar: IODDT types (T_ANA...) are unknown so not allowed for restore. |
| Device DDTs cannot be restored. | DevDDT: Device DDT types (T_...) are not allowed for restore. |

# Transferring current values

## At a Glance

The functions below apply to the following variables:

- located and unlocated variables
- for EFBs and DFBs:
  - public variables,
  - private variables
  - input parameters entered in the form of a value
  - output parameters entered in the form of a value

## Update Init Value with Current Value

The **Update Init value with Current Value** function is used to replace the initial values in the PLC of these variables with their current values if their **Save** attribute is checked.

When the initial values are updated in the PLC, a "*" is displayed in the status bar to indicate that the initial values in the PLC are different from the initial values displayed in the variable editor.

Use **PLC** → **Update Init Values with Current Values** to activate this function.

## Update Local Init Values with PLC Init Values

When the Initial values have been modified by the function **Update Init value with Current Value** or by using %S94 in the application, the **Update Local Init Values with PLC Init Values** function updates the initial values on the PC in the variable editor.

After using this function "*" is no longer displayed in the status bar.

Use **PLC** → **Update Local Init Values with PLC Init Values** to activate this function.

# Memory usage

## At a Glance

The memory usage function is used to view:

- the physical distribution of the PLC memory *(see Unity Pro, Program Languages and Structure, Reference Manual )* (internal memory and memory card)
- the space taken up in the memory by a project (data, program, configuration, system)

It can also be used to reorganize the memory where possible.

**NOTE:** The memory usage screen is not available in simulation mode. This screen is only available in standard mode when you have built the application.
If the application has been built and if it is in NOT BUILT state due to a program modification, the screen is accessible, but it corresponds to the application built previously. Modifications will be taken into account at the next build.

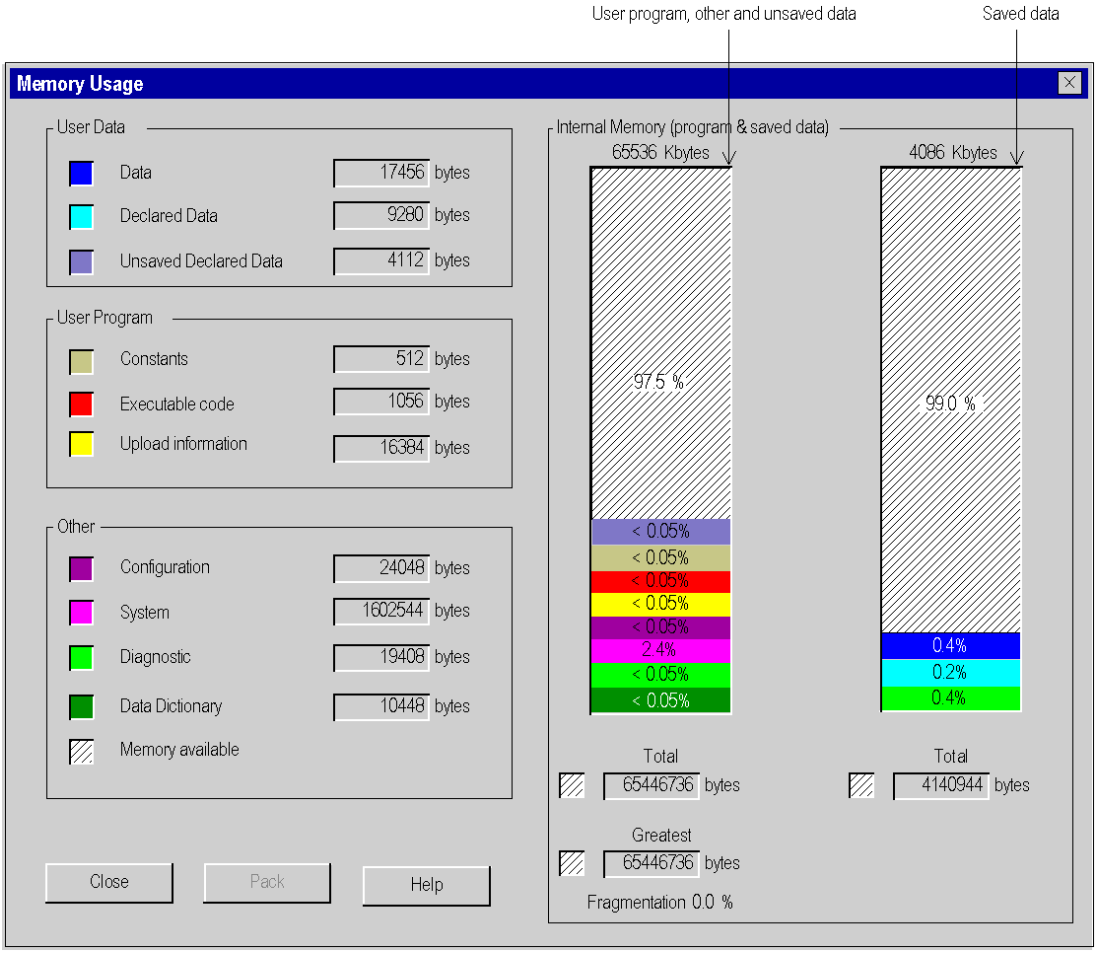
## Procedure

Perform these actions to access the memory usage details of the PLC:

| Step | Action |
|---|---|
| 1 | Activate the **PLC** → **Memory Consumption...** command, which displays the corresponding screen. The memory usage statistics of a project can only be accessed if you have generated its executable in advance. |
| 2 | If you want to reorganize the memory to optimize it, activate the **Pack** command. |

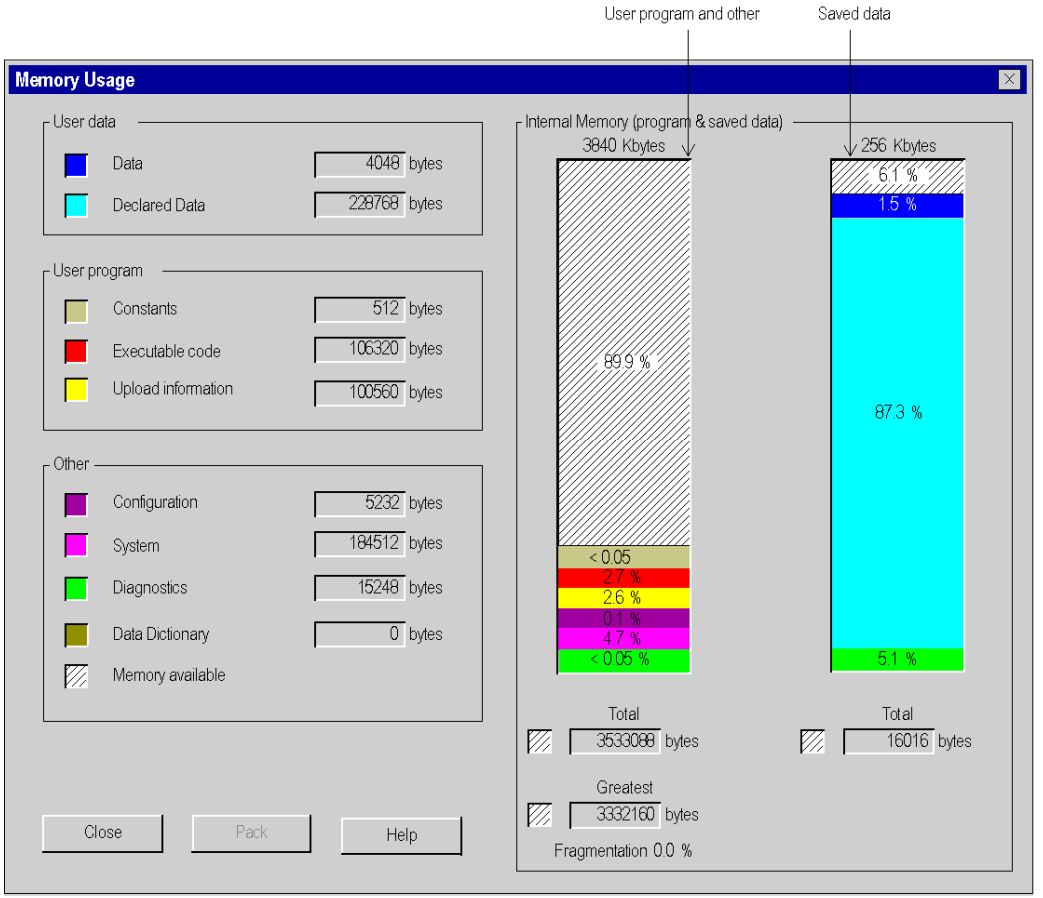
## Memory usage screen Modicon M580

The memory usage screen is shown below:



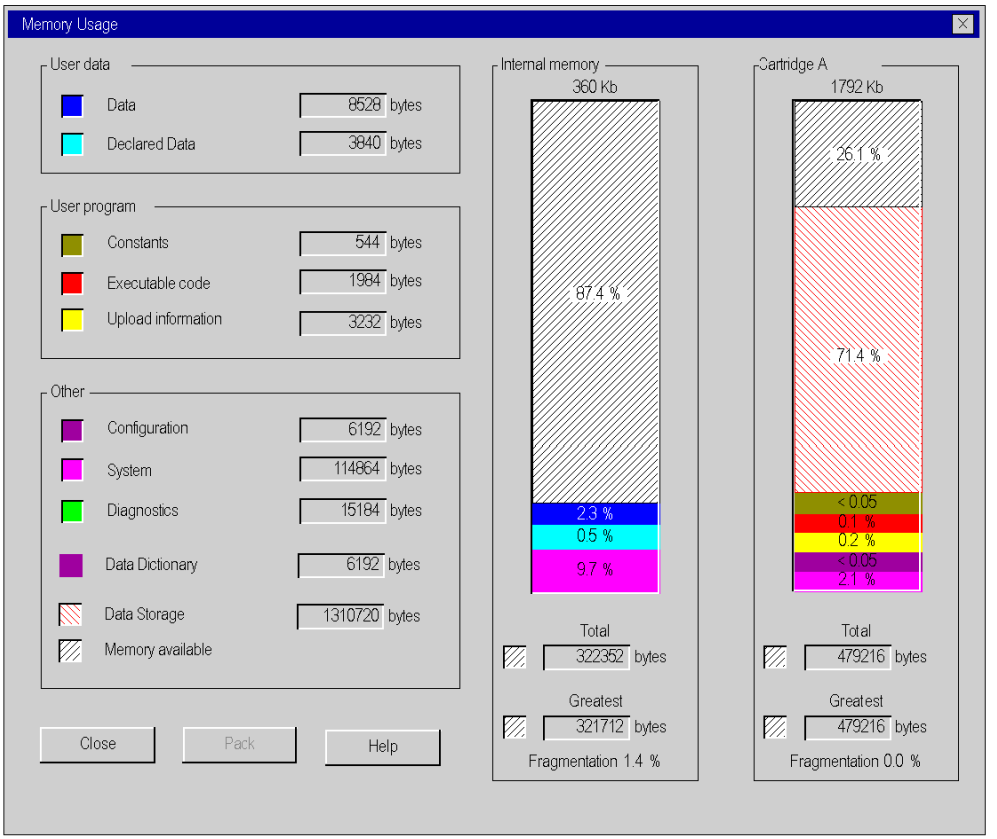
## Memory usage screen Modicon M340

The memory usage screen is shown below:



## Memory usage screen Premium/Quantum

The memory usage screen of a PLC with a memory card and data storage is shown below. If the PLC does not have a memory card, only the information relating to the internal memory is displayed.



## Description of the parameters

The following information fields are available:

| Parameter | Description |
|---|---|
| **User Data** | This field indicates the memory space (in words) taken up by user data (objects relating to configuration):<br>● **Data**: located data associated with the processor (%M, %MW, %S, %SW, etc.) or the input/output modules,<br>● **Declared Data**: unlocated data (declared in the data editor) saved after power cut.<br>● **Unsaved Declared Data**: unlocated data (declared in the data editor) not saved after power cut. |
| **User Program** | This field indicates the memory space (in words) taken up by the project program:<br>● **Constants**: static constants associated with the processor (%KW) and the input/output modules; initial data values,<br>● **Executable code**: executable code of the project program, EFs, EFBs and DFB types,<br>● **Upload information**: information for uploading a project (graphic code of languages, symbols, etc.). |
| **Other** | This field indicates the memory space (in words) taken up by other data relating to the configuration and the project structure:<br>● **Configuration**: other data relating to configuration (Page0 for a Quantum PLC, hardware configuration, software configuration),<br>● **System**: data used by the operating system (task stack, catalogs, etc.),<br>● **Diagnostic**: information relating to process or system diagnostics, diagnostics buffer,<br>● **Data Dictionary**: Dictionary of symbolized variables with their characteristic (address, type....)<br>● **Data Storage**: file and data storage (only for Premium and Quantum). |
| **Internal Memory** | This field shows the organization of the PLC's internal memory. It also indicates the memory space available (**Total**), the largest possible contiguous memory space (**Greatest**) and the level of fragmentation (due to online modifications). |
| **Cartridge A / Cartridge B** | Only for Premium and Quantum, this field shows the organization of the memory cards. It also indicates the memory space available (**Total**), the largest possible contiguous memory space (**Greatest**) and the level of fragmentation (due to online modifications). |
| **Pack** | This command is used to reorganize the memory structure. |

## Memory re-organization

Memory re-organization is activated using the **Pack** command *(see page 152)*.

Memory re-organization can be performed in online or offline mode (Even if the PLC is in Run or in Stop).

**NOTE:** Certain blocks cannot be moved in online mode. You will attain a lower level of fragmentation by re-organizing the memory in offline mode.

# Function Pack for Memory

## At a Glance

With the **Pack** command it is possible to reorganize the memory so as to optimize it.

To avoid certain detected errors (Internal Memory full or Cartridge Memory full) when you do a build after online modifications, the **Pack** command must be carried out before the online modifications.

**NOTE:** When the **MEM** status of the Status bar becomes **Red**, it is recommended to make a **Pack** command by clicking on the **Pack** button in the Memory usage screen *(see page 150)* of the PLC.

The necessity of **Packing** the internal (or cartridge) memory is evaluated by the PLC operating system and recommended when:

- the degree of fragmentation (due to many online modifications for instance) of the memory is becoming high,
- and/or the size of the largest contiguous block of available memory is becoming low compared to the total size of available memory.

**NOTE:** for Quantum PLCs, the **Pack** command can be used for OS version 2.3 or later.

## Elements that Require Large Amounts of Memory

Certain project elements and actions consume a large amount of memory:

- configuration with a Cartridge memory:
  - with a large section,
  - with several actions in the same SFC section.
- configuration with an internal memory:
  - with a large DFB instances,
- configuration with an internal memory in a Hot StandBy project:
  - several DFB instances,
  - adding or deleting DFB instances,
  - modifying an SFC section.

## Procedure

Perform the following actions to use the **Pack** command:

| Step | Action |
|---|---|
| 1 | Activate the **PLC → Memory Consumption:** command, which displays the corresponding screen. |
| 2 | In the Internal Memory or Cartridge area check the percentage values of:<br>● **Memory available**,<br>● **Fragmentation**,<br>● **Greatest / total of Memory available**. |
| 3 | Check if these values are in the ranges listed in the tables below. |
| 4 | If Yes, click on the **Pack** button. |
| 5 | Above the **Pack** button, if the warning **Memory optimization is not terminated, please continue to pack** appears, the **Pack** button must be clicked again. |
| 6 | The **Pack** function is finished when:<br>● the **Pack** button is grayed,<br>● the **Fragmentation** field is equal to 0,<br>● the **Greatest** value is near the **Total Memory available** value. |

## How the PLC detects it's recommended to make a Pack command for Internal Memory

The following table shows how the PLC detects it's recommended to make a **Pack** command, regarding the field values in the Internal memory area:

| If the Percentage of Memory available is, | then if the Percentage of Fragmentation is, | OR if Greatest (1) / Total Available Memory is: |
|---|---|---|
| >25% | >15% | <50% |
| [15% ... 25%] | >10% | <80% |
| <15% | >5% | <90% |
| **(1) Greatest** is the Largest free contiguous block of memory available, this information is given in the Memory usage screen. | | |

## How the PLC detects it's recommended to make a Pack command for Cartridge Memory

The following table shows how the PLC detects it's recommended to make a **Pack** command, regarding the field values in the Cartridge memory area:

| If the Percentage of Memory available is, | then if the Percentage of Fragmentation is, | OR if Greatest (1) / Total Available Memory is: |
|---|---|---|
| >25% | >10% | <70% |
| [15% ... 25%] | >5% | <90% |
| <15% | >3% | <95% |
| **(1) Greatest** is the Largest free contiguous block of memory available, this information is given in the Memory usage screen. | | |

# Memory Card Access for Modicon M340

## Overview

BMX RMS ••••• memory cards are used to save projects, Web pages, and data in general.

They are mainly used when inserted into a PLC. They may also be read with an SDcard drive or reader.

## Memory Card in the PLC

When the memory card is in the Modicon M340, it may be accessed as follows:

- with the PLC, automatically
- with the Unity Pro commands *(see page 132)* **PLC → Project Backup... → ....**
- by programming with the Memory Card File Management EFBs (See Memory Card File Management in the System library)
- with an FTP *(see page 156)* client (for processors with an Ethernet connection) to manage files on the memory card

## Memory Card in an SDcard Drive or Reader

When the memory card is inserted into an SDcard drive or reader, it may be accessed as a data storage medium (for example, a USB key or hard drive). The files on the memory card may thus be viewed.

**NOTE:** In order to read the memory card in an SDcard drive or reader, install the Reliance driver first. This driver is available on Unity Loader CD-ROM.

## Tree Structure of Files on the Memory Card

When the memory card is used in an SDcard drive or reader, or via FTP, its contents are accessible with a file explorer. The tree structure shown includes three directories:

- *DataStorage*, which includes all data files from Memory Card File Management EFBs *(see Unity Pro, System, Block Library)*
- *Firmware*, which includes all files used by Unity Loader
- *Web*, which includes all Web pages *(see Modicon M340 for Ethernet, Communications Modules and Processors, User Manual)*

It is also possible to create directories for personal files.

**NOTE:** The memory zone for project files is not accessible by users.

# FTP and Memory Card for Modicon M340

## Overview

With an FTP client, it is possible to transfer files between the memory card in the Modicon M340 PLC (with an embedded Ethernet port connection), a destination, and/or a source such as a hard drive or server.

Any FTP client installed on your computer may be used. We recommend using Filezilla or Microsoft Explorer.

You can transfer:

- data files used by the memory card file management EFBs
- web pages used by the HTTP server
- personal files that are useful for your project

**NOTE:** Downloading write-protected files to the memory card can prevent the Unity loader from properly upgrading the module. Some FTP clients (for example, the Windows Explorer client) cannot remove write-protected files from the card. You can delete write-protected files from the card with some FTP clients (for instance, FileZilla) that are available for free download over the Internet.

## Using FTP

The following table describes how to use FTP with Internet Explorer.

| Step | Action |
|---|---|
| 1 | Open Internet Explorer. |
| 2 | Input the FTP address in the Address field.<br>Example: ftp://login:password@FTP_Server_IP_Address<br>● Login: datastorage<br>● Password: datadownload<br>**Result**: The contents of the memory card are displayed. |
| 3 | Copy and/or paste the files between the memory card and your desired location. |

The following modules have a **time out value of 15 minutes** and a **maximum number of eight sessions**:

- BMX P34 20•0
- BMX NOE 01•0
- BMX NOC 0401
- BMX NOR 0200
- BME P58 ••••
- BME NOC 03•1
- TSX P57 •634
- TSX ETY •10••
- TSX ETY PORT
- 140 CPU 651 •0
- 140 NOE 771 ••
- 140 NOC 78• 00

# Project Management with DTMs

## At a Glance

The Unity Pro Project and Archive files can be saved and opened even if DTMs are not installed in the DTM Hardware Catalog *(see page 257)*.

Building a complete Unity Pro project requires that all DTMs in the project be installed.

## Opening STU and STA Files

Opening the *.stu Project file and *.sta Archive files restores the DTM topology tree and device configurations.

If there are DTMs in the restored project that are not installed on the host PC, there is no warning message. Services such as Rebuild all or Check devices *(see page 274)* must be used to check for un-installed DTMs.

## Saving STU and STA Files

The DTM topology tree and device configurations are saved in these files.

## ZEF Files

The DTM topology tree and device configurations is saved and restored in the *.zef import/export files.

## XEF Files

The DTM topology tree and device configurations cannot be saved nor restored in the *.xef import/export files.

## Complete Build Management

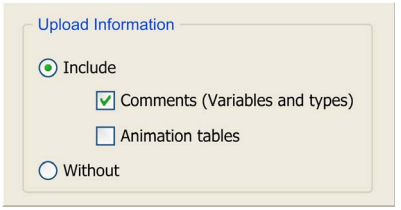
Rebuild all project *(see page 90)* (DTM modifications or not) requires that all DTMs in the project be installed on the PC because the **Check devices** service is part a complete build.

If any DTMs are not found, a detected error is displayed in 2 places:

- The Build tab in the Output WIndow
- DTM topology tree

## Partial Build Management

The user can choose not to include the DTM configuration in a PCL binary:




A partial project build that does not take into account the DTM configurations does not call the **Check devices** service, therefore, this build does not need to have all the DTMs installed on the PC.

## Upload Project

The Unity Pro Transfer Project from PLC *(see page 93)* (Upload project) function uploads the current DTM topology tree and the device configurations stored in the PLC (but not the information in the Profibus Remote Bus Master and the devices). This function is not the same as the Load data from device DTM *(see page 277)* service.

Some DTMs have a Compare function that allows comparing the configuration in the device to the configuration in the PLC.

## Download Project

To use the Unity Pro **Transfer Project to PLC** (Download project) function with DTMs:

| Step | Action |
|---|---|
| 1 | Construct a DTM topology tree. |
| 2 | Use the Store to device *(see page 277)* service on each DTM that has a physical device on the network. |
| 3 | Call the **Transfer Project to PLC** function to download the Unity Pro project that includes a DTM topology tree and device configurations. |

# Chapter 5
# Project browser

## Subject of this chapter

This chapter introduces the Unity Pro browser that allows you to move around your project, via the structure view or function view of its directory tree.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 5.1 | Introduction to the project browser | 162 |
| 5.2 | Introduction to the various directories of the project browser | 193 |
| 5.3 | Summaries of the utilities associated with the structure view and function view | 215 |

# Section 5.1
## Introduction to the project browser

### About this sub-chapter

This sub-chapter provides general information on the project browser.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction to the project browser | 163 |
| Structural view and functional view of the project | 165 |
| The Zoom In and Zoom Out commands | 167 |
| The commands Contract all and Expand all | 170 |
| The Goto command | 174 |
| User Directory | 177 |
| Project Properties | 179 |
| Application Protection | 180 |
| Firmware Protection | 184 |
| Section and Subroutine Protection | 186 |
| Data Storage Protection | 188 |
| Loss of Password | 190 |

# Introduction to the project browser

## General points

The project browser allows you to display the contents of a Unity Pro project and to move around its various elements: configuration, data, program, etc. To do this you can display your project in 2 different ways:

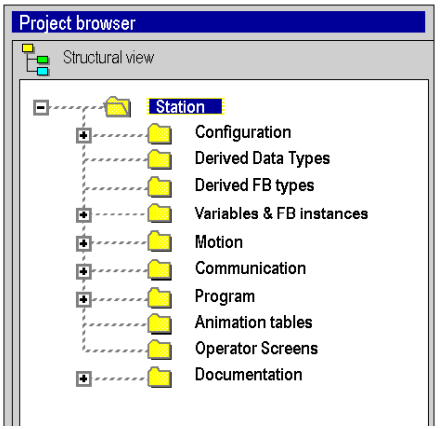
- **Structural View**,
- **Functional View**.

## Structural view

The structural view displays the directory tree of the Unity Pro project and allows you to navigate within this with direct access:

- to the configuration,
- to the DDT and DFB types,
- to the variables (EDT, DDT, DFB and EFB instances),
- to motion functions,
- to communication functions,
- to the program,
- to the animation tables,
- to the operator screens,
- to the project documentation (title page, general information).

The following illustration shows the structural view of a project:



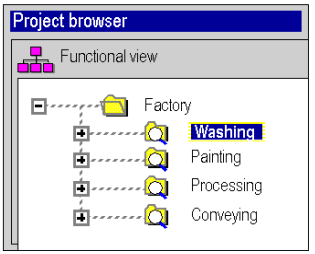

By default, the browser displays the second level of the directory tree. To access the other levels, you need to expand the directories.

**NOTE:** The default project directory name is **Station**. You can change this name by accessing the project properties dialog box using the **Properties** command from the contextual menu.

## Functional view

The functional view displays the directory tree of the project, broken down into functional modules *(see page 221)*. This breakdown does not take into account the order of execution of the program by the PLC.

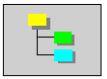
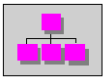
The following illustration shows the functional view of a project:




By default, the browser displays the first level of the directory tree. To access the other levels, you need to expand the directories.

## Navigating between the functional view and structural view.

In the toolbar of the project browser you will find the following icons that allow you to display the various project views:

| | |
|---|---|
|  | Allows you to display the structural view of the project. |
|  | Allows you to display the functional view of the project. |
|  | Allows you to display the structural view and the functional view in series *(see page 165)*. |
|  | Allows you to display the structural view and the functional view in parallel *(see page 166)*. |

# Structural view and functional view of the project

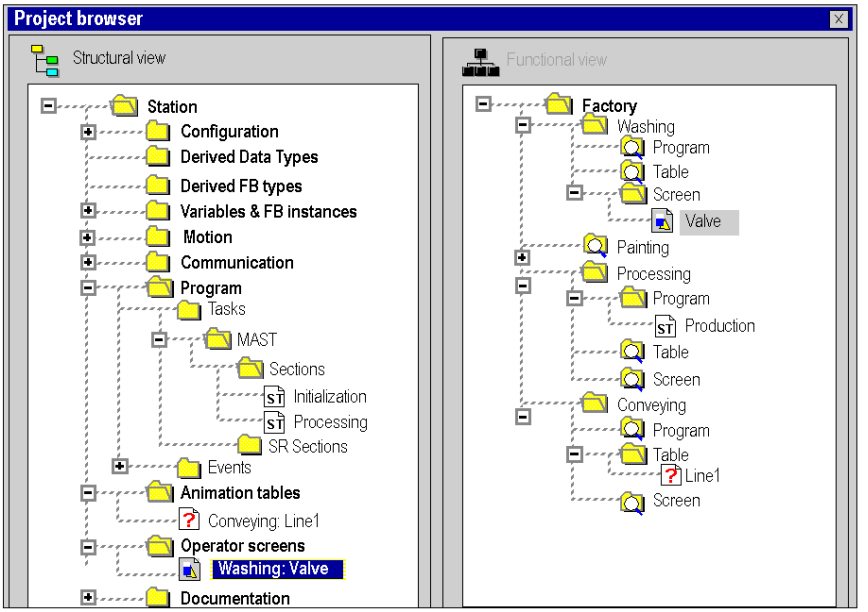
## At a Glance

The project browser allows you to simultaneously display the structural view and functional view of the project. You can choose to display these views:

- in series. Both views are displayed, one next to the other,
- in parallel. Both views are displayed, one above the other.

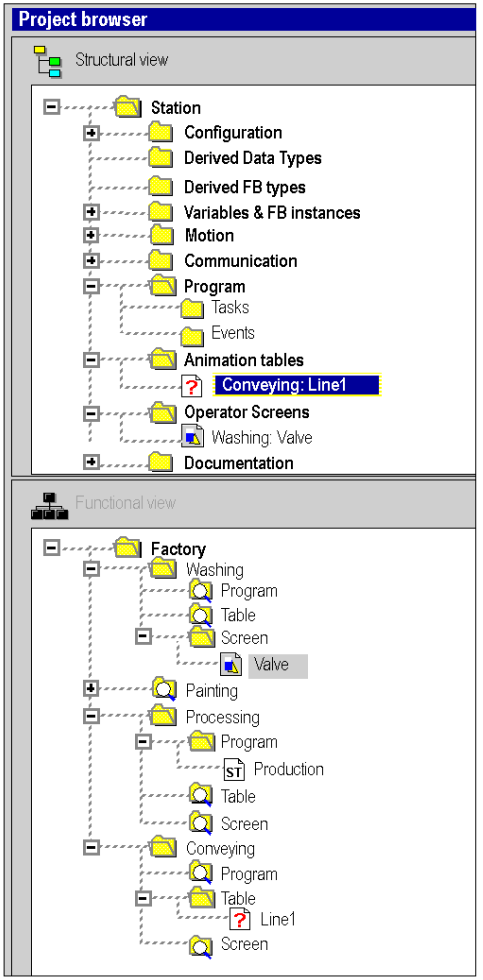
## Display of both views in series

The following illustration shows a series display of the structural view and functional view:



## Display of both views in parallel

The following illustration shows a parallel display of the structure and functional views:



# The Zoom In and Zoom Out commands

## At a Glance

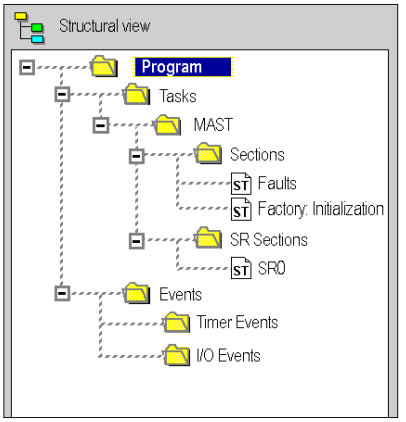
In the functional view and structural view of the project browser, you can display a specific directory tree level by using the **Zoom In** command. You can apply this command in a level below the one you are in. This command is no longer available once you cannot expand the tree structure any further.

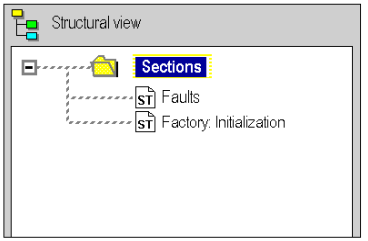
The **Zoom Out** command allows you to go back to the previous display of the project browser.

**NOTE:** The following steps show how to use the **Zoom In** and **Zoom Out** commands with a structural view. These steps can also be used with a functional view.

## Displaying a specific level of the directory tree

The following table shows you how to use the **Zoom in** command in the structural view:

| Step | Action |
|---|---|
| 1 | In the project browser select the directory you wish to display (for example **Program**).<br>**Result**: The element selected appears in reverse video. |
| 2 | Activate the **Zoom In** command from the **Display** menu or from the contextual menu.<br>**Result**: Only the directory selected appears in the structural view (for example **Program**):<br>  |

| Step | Action |
|---|---|
| 3 | From the directory (for example **Program**) select the directory (for example **Sections**) that you wish to display.<br>**Result**: The element selected appears in reverse video. |
| 4 | Activate the **Zoom In** command from the **Display** menu or from the contextual menu.<br>**Result**: Only the directory selected appears in the structural view (for example **Sections**):<br>  |

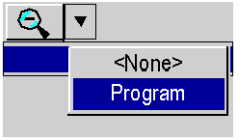
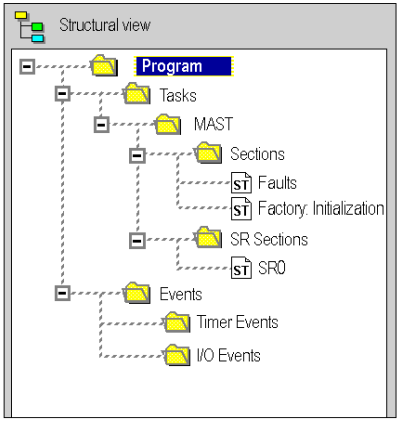
## Returning to the structural view of the project

The following table shows you how to use the **Zoom Out** command in the structural view:

| Step | Action |
|---|---|
| 1 | In the project browser select the directory from the structural view.<br>**Result**: The element selected appears in reverse video. |
| 2 | Activate the **Zoom Out** command from the **Display** menu or from the contextual menu or by clicking on the  icon.<br>**Result**: The structural view appears along with the directory tree of the project. |

## Returning to directory tree level

The following table shows you how to use **Zoom Out** command in a lower level of the directory tree of the structural view:

| Step | Action |
|---|---|
| 1 | In the project browser select the directory from the structural view (for example **Sections**).<br>**Result**: The element selected appears in reverse video. |
| 2 | Expand the menu using the arrow to the right of the  icon.<br>  <br><br>**Result**: The menu with the choice of different views appears. |
| 3 | In the menu (for example **Program**) select the structural view to be displayed.<br>**Result**: The directory (for example **Program**) appears in the structural view:<br>  <br><br>Selecting **None** displays the entire structural view of the project. |

# The commands Contract all and Expand all

## At a Glance

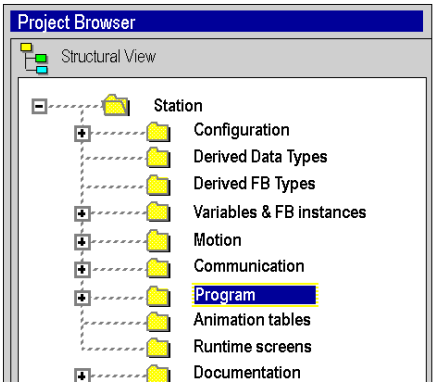
In the functional view and structural view of the project browser, you can display a specific directory tree level and its sub-directories by using the **Expand all** command.

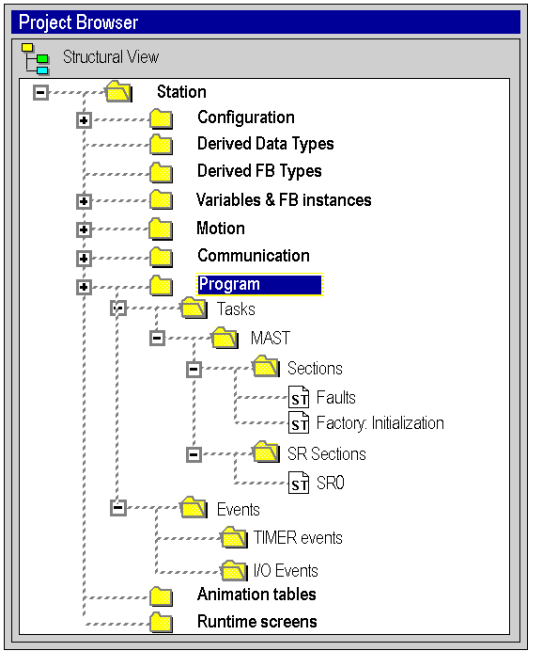
The command **Contract all** allows you to contract an entire directory.

**NOTE:** The following steps show how to use the **Contract all** and **Expand all** commands with a structural view. These steps can also be used with a functional view.

## How to expand a directory

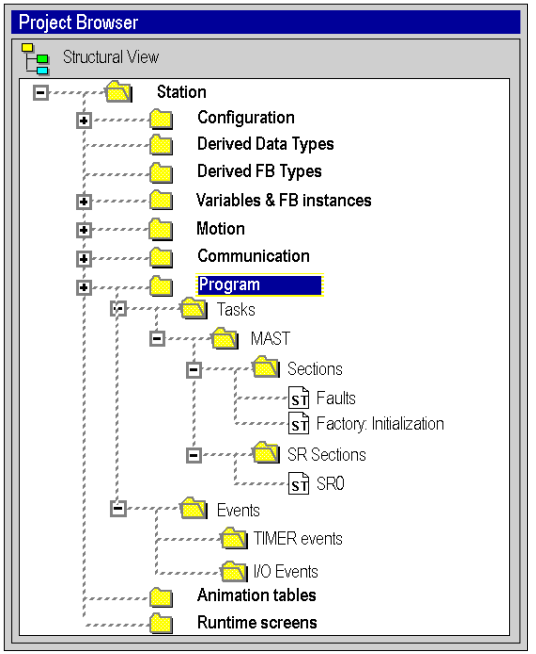
The following table shows you how to use the **Expand all** command in the structural view:

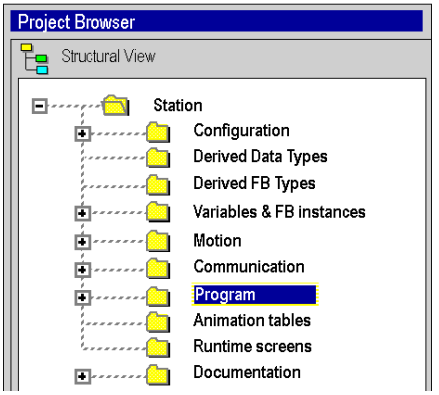
| Step | Action |
|---|---|
| 1 | In the project browser select the directory you wish to expand (for example **Program**).<br>**Result**: the element selected appears in reverse video:<br>  |

| Step | Action |
|---|---|
| 2 | Activate the **Expand all** command from the **Display** menu or from the contextual menu.<br>**Result**: the selected directory is expanded completely:<br>  |

## How to contract a directory

The following table shows you how to use the **Contract all** command in the structural view:

| Step | Action |
|---|---|
| 1 | In the project browser select the directory you wish to contract (for example **Program**).<br>**Result**: the element selected appears in reverse video:<br>  |

| Step | Action |
|---|---|
| 2 | Activate the **Contract all** command from the **Display** menu or from the contextual menu.<br>**Result**: the selected directory is contracted:<br>  |

# The Goto command

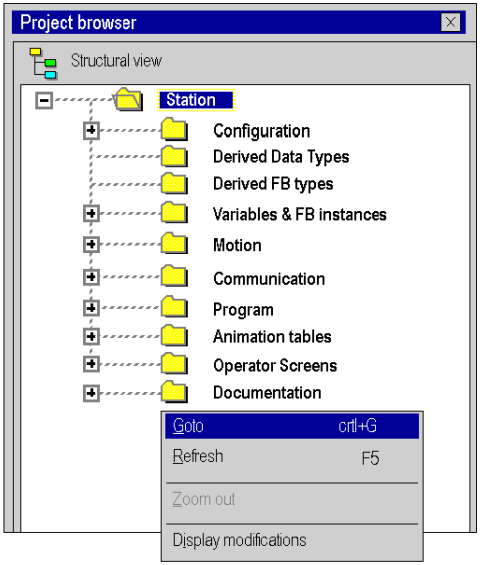
## At a Glance

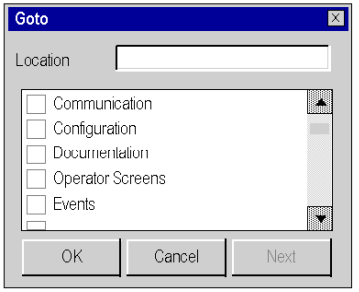
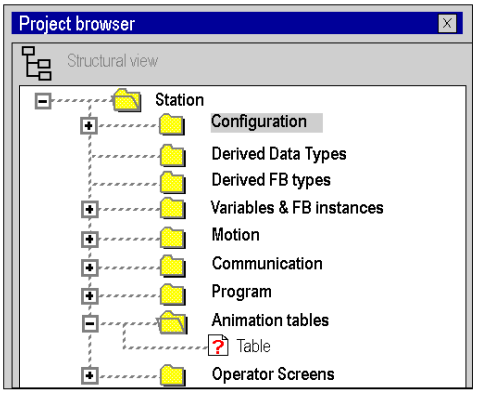
The **Goto** command allows you to access a desired location in the project browser.

In the functional view or structural view, the levels are not always expanded. The **Goto** command allows you to go directly to what you are looking for.

## Finding an element

The following table shows you how to use the **Goto** command in the structural view of the browser:

| Step | Action |
|---|---|
| 1 | Click the right mouse button on the background of the structural view.<br>**Result**: The contextual menu appears.<br>  |

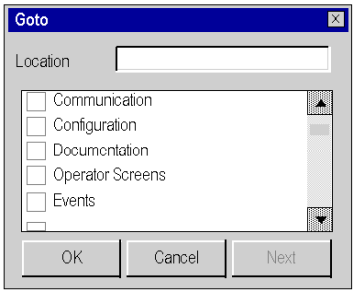
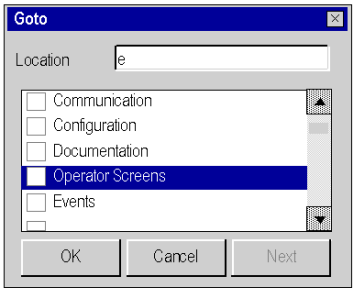
| Step | Action |
|---|---|
| 2 | Select the **Goto** command from the menu.<br>**Result**: The **Go to** window appears.<br>  |
| 3 | In the window, double-click the element you want to go to.<br>**Result**: The element appears shaded in the structural view (Configuration in the example).<br>  |
| 4 | In the **Goto** window confirm your choice with **OK**. |

## Elements with the same name

Where one or more elements have the same name (examples: sections). In the **Go to** window the element appears with a marker. ☑ . The **Next** command in the **Go to** window is unshaded. This allows you to pinpoint one by one the different elements with the same name when the **Go to** window is active.

## Fast access to an element

In order to access an element in the window more quickly in the **Go to** window, you can use the **Location** function. Typing in the first letter of the element gives you access to the elements beginning with these letters. The following table shows you how to use the **Location** function in the structural view of the browser:

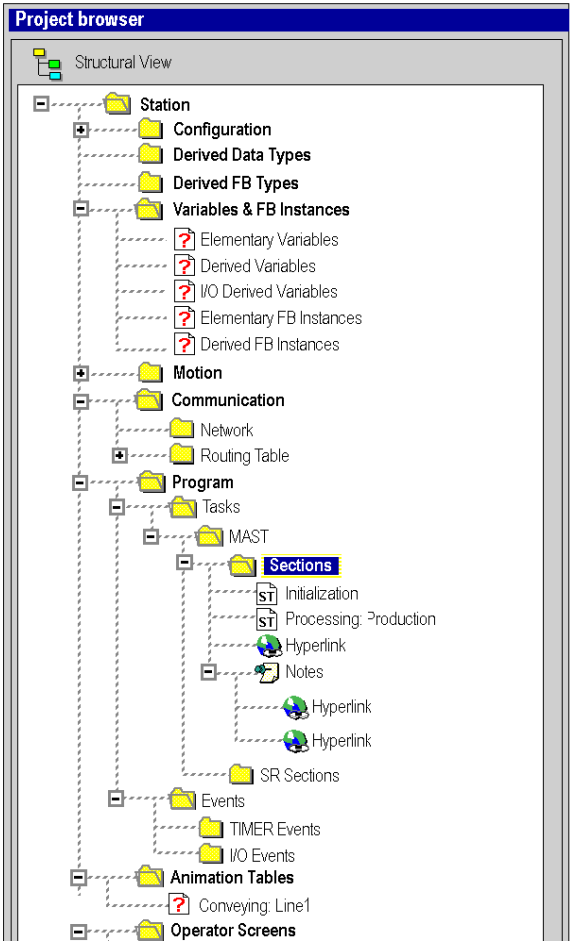
| Step | Action |
|---|---|
| 1 | Click the right mouse button on the background of the structural view. |
| 2 | Select the **Goto** command from the menu.<br>**Result**: The **Go to** window appears.<br>  |
| 3 | In the **Location** field type the first letter of the element you want to go to.<br>**Result**: The element appears in reverse color in the **Go to** window.<br>  |
| 4 | In the **Go to** window confirm your choice with **OK**. |

# User Directory

## At a Glance

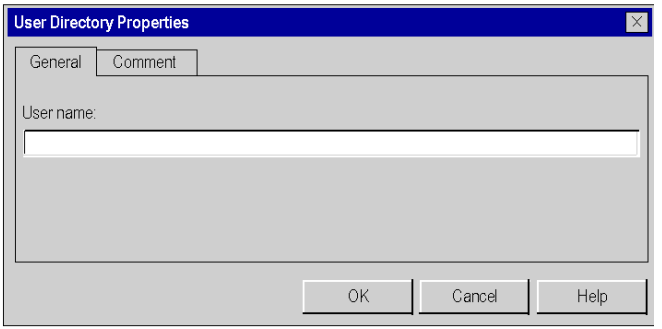
In the **Station** directory and in all the directories of the project browser, you can create hyperlinks *(see page 1817)* and user directories for grouping of hyperlinks.

**NOTE:** You cannot nest a user directory in a user directory and a hyperlink in a hyperlink.
The following illustration shows a hyperlink and a user directory in the directory **Program** :



## Creating a User Directory

The following table shows you how to create a user directory in the structural view of the browser:

| Step | Action |
|---|---|
| 1 | In the project browser select the directory in which you wish to create a user directory.<br>**Result**: The element selected appears in reverse video. |
| 2 | Select the **Add User Directory.. command from the popup menu**..<br>**Result**: the **User Directory Properties** window appears:<br>  |
| 3 | Give the user directory a name in the **General** tab. |
| 4 | In the **Comment** tab enter the comment associated with the user directory. |
| 5 | Click **OK** to confirm your choice.<br>**Result**: The user directory appears in the project browser. |
| 6 | You can now create hyperlinks *(see page 1817)* in the directory provided for this purpose. |

# Project Properties

## At a Glance

Whichever view is displayed, you can access the project properties by using the **Properties** command from the contextual menu (which can be reached by right-clicking the mouse on the **Project** folder). This gives you access to a dialog box which allows you to:

- access information on the current project,
- define certain parameters of the project.

## Description of the Tabs

The dialog box has 4 tabs allowing the actions given below:

| Tab | Description |
|---|---|
| **General** | **Name**: allows you to define the name of the project.<br>**Type**: indicates the type of Unity Pro software used.<br>**FFB Library version**: indicates the Libset version used for your current project. |
| **Protection** | Password setting applies to the following:<br>● **Application**:(1) application password settings *(see page 180)*<br>● **Sections**: sections password settings *(see page 186)*<br>● **Firmware**(2): firmware password settings *(see page 184)*<br>● **Data Storage**(3): data storage password settings *(see page 188)* |
| **Identification** | Allows you to identify the project:<br>● current version with the option of automatic increment<br>● creation date<br>● generation date<br>● Last rebuild all<br>● Last partial build |
| **Comment** | Allows a comment to be associated with the project. |

**1** Except for Modicon Momentum.
**2** Only for Modicon M580 CPUs with OS version ≥ 2.00, BME NOC 03••, and BME CXM 0100 modules.
**3** Only for Modicon M580 CPUs.

**NOTE:** By default, the dialog box is called **Properties of Project**. If you give the project another name (**myproject** for example), this dialog box is called **Properties of myproject**.

# Application Protection

## Overview

CPU application protection by a password prevents unwanted application modification, download, or opening (.STU and .STA files). The password is stored encrypted in the application.

## Password

An application is not password protected by default.

It is possible to create or change a password at any time.

The password is case-sensitive and it must have a size from 8 to 16 alphanumeric characters. The password robustness is increased when it contains a mix of upper and lower case, alphabetical, numerical, and special characters.

## Creating a Password

Procedure for creating the application protection password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Application** field, click **Change password ...**.<br>**Result**: The **Modify Password** window appears. |
| 5 | Enter the new password in the **Entry** field. |
| 6 | Enter the confirmation of the new password in the **Confirmation** field. |
| 7 | Click **OK** to confirm. |
| 8 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Changing the Password

Procedure for changing the application protection password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Application** field, click **Change password ...**.<br>**Result**: The **Modify Password** window appears. |
| 5 | Enter previous password in the **Old password** field. |
| 6 | Enter the new password in the **Entry** field. |
| 7 | Enter the confirmation of the new password in the **Confirmation** field. |
| 8 | Click **OK** to confirm. |
| 9 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Deleting the Password

Procedure for clearing the application protection password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Application** field, click **Clear password...**.<br>**Result**: The **Password** window appears. |
| 5 | Enter the password in the **Password** field. |
| 6 | Click **OK** to confirm. |
| 7 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Auto-Lock Feature

There is an optional feature to limit access to the Unity Pro software programming tool after a configured time of inactivity. After that time of inactivity a dialog box is displayed and prompts to enter the application password.

Behind the modal dialog box, the current opened editors remain opened in the same position. That means that anybody can read the current content of the Unity Pro windows but cannot continue to work with Unity Pro.

You can activate the auto-lock feature with the check box **Auto-lock** and select the time-out for the time of inactivity via **Minutes before lock**.

The default values are:
- **Auto-lock** is not activated
- **Minutes before lock** is set to 10 minutes (possible values: 1...999 minutes)

## Password Request Condition

Open an application (project):

| **Password Management** | |
|---|---|
| When an application file is opened, an **Application Password** dialog box opens. | |
| Enter the password. | |
| Click **OK**. | If the password is correct, the application opens. |
| | If the password is wrong, a dialog box pops up to indicate it and a new **Application Password** dialog box opens. |
| If you click **Cancel**, the application is not opened | |

Unity Pro software auto-lock:

| **Password Management** | |
|---|---|
| When auto-lock time is elapsed, an **Application Password** dialog box opens: | |
| Enter the password. | |
| Click **OK**. | If the password is correct, Unity Pro becomes active again. |
| | If the password is wrong, a dialog box pops up to indicate it and a new **Application Password** dialog box opens. |
| If you click **Close**, the application is closed without being saved. | |

CPU application auto-lock:

| **Password Management** | |
|---|---|
| When auto-lock time is elapsed, an **Application Password** dialog box opens: | |
| Enter the password. | |
| Click **OK**. | If the password is correct, the connection between Unity Pro and the CPU becomes active again. |
| | If the password is wrong, a dialog box pops up to indicate it and a new **Application Password** dialog box opens.<br>**NOTE:** The password must be entered to have the connection and the request cannot be canceled. |

CPU connection to Unity Pro:

| **Password Management** | |
|---|---|
| On connection, if Unity Pro software application and the CPU application are not equal, an **Application Password** dialog box opens: | |
| Enter the password. | |
| Click **OK**. | If the password is correct, the connection is established. |
| | If the password is wrong, a dialog box pops up to indicate it and a new **Application Password** dialog box opens. |
| If you click **Cancel**, the connection is not established. | |
| **NOTE:** On connection, if Unity Pro software application and the CPU applications are equal, there is no password request. If no password has been initially entered (left empty on project creation), click **OK** to establish the connection on password prompt. | |

**NOTE:** For each new password request, the elapsed time increases between the wrong password notification and a new **Application Password** dialog box prompt (up to 1 hour).

**NOTE:** In case of password loss, contact Schneider Electric support *(see page 190)*.

# Firmware Protection

## Overview

Firmware protection by a password helps prevent unwanted access to the module firmware via FTP.

**NOTE:** Firmware password is implemented from Unity Pro V10.0. An application created with Unity Pro V10.0 cannot be opened with an earlier version of Unity Pro.

## Password

The firmware is password protected by default with the following password: **`fwdownload`**.

It is possible to change a password at any time.

The password is case-sensitive and contains 8 to 16 alphanumeric characters. The password robustness is increased when it contains a mix of upper and lower case, alphabetical, numerical, and special characters.

**NOTE:** When importing a ZEF file, the firmware password of the module is set to its default value.

## Changing the Password

**NOTE:** Firmware default password: **`fwdownload`**

Procedure for changing the firmware protection password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Firmware** field, click **Change password ...**.<br>**Result**: The **Modify Password** window appears. |
| 5 | Enter previous password in the **Old password** field. |
| 6 | Enter the new password in the **Entry** field. |
| 7 | Enter the confirmation of the new password in the **Confirmation** field. |
| 8 | Click **OK** to confirm. |
| 9 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Resetting the Password

Resetting the password assigns its default value to the firmware password (**fwdownload**) once the current password is confirmed. Proceed as follows:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Firmware** field, click **Reset password...**.<br>**Result**: The **Password** window appears. |
| 5 | Enter current password in the **Password** field. |
| 6 | Click **OK** to confirm. |
| 7 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes. The new password is the default password: **fwdownload**.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

# Section and Subroutine Protection

## At a Glance

The section protection function is accessible from the **Properties** screen of the project in offline mode.

This function is used to protect the program sections.

**NOTE:** The section protection is not active as long as the protection has not been activated in the project.

**NOTE:**
The project protection is effective to the marked sections only. This does not prevent from:
- Connecting to the CPU
- Uploading application from the CPU
- Changing the configuration
- Adding new sections
- Changing the logic in a new (not protected) section

## Activating Protection and Creating Password

Procedure for activating the protection of sections and creating the password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Sections** field, activate the protection by checking the **Protection active** box.<br>**Result**: The **Modify Password** dialog box appears: |
| 5 | Enter a password in the **Entry** field. |
| 6 | Enter the confirmation of the password in the **Confirmation** field. |
| 7 | Select **Crypted** check box if an enhanced password protection is required.<br>**NOTE:** A project with a crypted password cannot be edited with a Unity Pro version lower than 4.1. |
| 8 | Click **OK** to confirm. |
| 9 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Notes

If a section *(see page 467)* is configured with a protection (read or read/write), when protection has been activated this will be indicated by a locked padlock at the section level.

If the section is configured with a protection but the protection is disabled, an open padlock is displayed at the section level.

## Changing the Password

Procedure for changing the project sections protection password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Sections** field, click **Change password ...**.<br>**Result**: The **Modify Password** dialog box appears: |
| 5 | Enter previous password in the **Old password** field. |
| 6 | Enter the new password in the **Entry** field. |
| 7 | Enter the confirmation of the new password in the **Confirmation** field. |
| 8 | Select **Crypted** check box if an enhanced password protection is required.<br>**NOTE:** A project with a crypted password cannot be edited with a Unity Pro version lower than 4.1. |
| 9 | Click **OK** to confirm. |
| 10 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Deleting the Password

Procedure for deleting the project sections protection password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Sections** field, click **Clear password...**.<br>**Result**: The **Access control** dialog box appears: |
| 5 | Enter the previous password in the **Password** field. |
| 6 | Click **OK** to confirm. |
| 7 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

# Data Storage Protection

## Overview

Data storage protection by a password helps prevent unwanted access to the data storage zone of the SD memory card (if a valid card is inserted in the CPU).

## Password

The data storage area is password protected by default with the following password: **`datadownload`**.

It is possible to change a password at any time.

The password is case-sensitive and it must have a size from 8 to 16 alphanumeric characters. The password robustness is increased when it contains a mix of upper and lower case, alphabetical, numerical, and special characters.

**NOTE:** When importing a ZEF file, the data storage password of the application is set to its default value.

## Changing the Password

**NOTE:** Data storage default password: **`datadownload`**

Procedure for changing the data storage protection password:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Data Storage** field, click **Change password ...**.<br>**Result**: The **Modify Password** window appears. |
| 5 | Enter previous password in the **Old password** field. |
| 6 | Enter the new password in the **Entry** field. |
| 7 | Enter the confirmation of the new password in the **Confirmation** field. |
| 8 | Click **OK** to confirm. |
| 9 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Resetting the Password

Resetting the password assigns its default value to the data storage password (**datadownload**) once the current password is confirmed. Proceed as follows:

| Step | Action |
|---|---|
| 1 | In the project browser right-click **Project**. |
| 2 | Select **Properties** command from the popup menu.<br>**Result**: The **Properties of Project** window appears. |
| 3 | Select **Protection** tab. |
| 4 | In the **Data Storage** field, click **Reset password...**.<br>**Result**: The **Password** window appears. |
| 5 | Enter current password in the **Password** field. |
| 6 | Click **OK** to confirm. |
| 7 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes. The new password is the default password: **datadownload**.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

# Loss of Password

## Overview

If you forget your password, proceed as indicated in the following procedures and contact Schneider Electric support.

## Unity Pro Application Password

Schneider Electric support needs a number displayed from the **Password** dialog box reached in following conditions:

- At open time, select the application and the **Password** dialog box is displayed.
- At auto-lock time, the **Password** dialog box is displayed. If you do not remember the password, select **Close**. Open the application again and the **Password** dialog box is displayed.
  **NOTE:** When the application is closed without entering a password after an auto-lock, all modifications are lost.

Procedure for resetting the application password:

| Step | Action |
|---|---|
| 1 | **Condition:** The **Password** dialog box is displayed. |
| 2 | Press `SHIFT+F2`.<br>**Result:** A grayed number is displayed in the right side of the **Password** dialog box. |
| 3 | Give this number to Schneider Electric support. |
| 4 | Receive the generated password from Schneider Electric support.<br>**NOTE:** The password is a temporary password, available as long as you do not modify the application. |
| 5 | Enter this password. |
| 6 | Modify the password (old password = password provided by Schneider Electric support). |
| 7 | Click **Build** → **Build Changes**. |
| 8 | **Save** the application. |

## CPU Application Password

Procedure for resetting the CPU application password if the respective **.STU* file is available:

| Step | Action |
|---|---|
| 1 | Open the respective **.STU* file. |
| 2 | When the password dialog box is displayed press `SHIFT+F2`.<br>**Result:** A grayed number is displayed in the right side of the **Password** dialog box. |
| 3 | Give this number to Schneider Electric support. |
| 4 | Receive the generated password from Schneider Electric support.<br>**Note:** The password is a temporary password, available as long as you do not modify the application. |
| 5 | Enter this password. |
| 6 | Modify the password (old password = password provided by Schneider Electric support). |
| 7 | **Connect** to the PLC. |
| 8 | Click **Build** → **Build Changes**. |
| 9 | **Save** the application. |

Procedure for resetting the CPU application password if the respective **.STU* file is not available:

| Step | Action |
|---|---|
| 1 | **Condition:** At connection time, the **Password** dialog box is displayed. |
| 2 | Press `SHIFT+F2`.<br>**Result:** A grayed number is displayed in the right side of the **Password** dialog box. |
| 3 | Give this number to Schneider Electric support. |
| 4 | Receive the generated password from Schneider Electric support.<br>**Note:** The password provided by Schneider Electric support is a temporary password, available as long as you do not modify the application. |
| 5 | Enter this password. |
| 6 | Upload the application from CPU. |
| 7 | **Save** the application. |
| 8 | Modify the password (old password = the one provided by Schneider Electric support). |
| 9 | Click **Build** → **Build Changes**. |
| 10 | **Save** the application. |

## Firmware Password

Schneider Electric support needs a number displayed from the **Password** dialog box reached in following condition:

- In **Properties of Project** → **Protection** → **Firmware** field, click **Reset password...** and the **Password** dialog box is displayed.

Procedure for resetting the firmware password:

| Step | Action |
|---|---|
| 1 | **Condition:** The **Password** dialog box is displayed. |
| 2 | Press `SHIFT+F2`.<br>**Result:** A grayed number is displayed in the right side of the **Password** dialog box. |
| 3 | Give this number to Schneider Electric support. |
| 4 | Receive the generated password from Schneider Electric support.<br>**Note:** The password is a temporary password, available as long as you do not modify the application. |
| 5 | Enter this password. |
| 6 | Modify the password (old password = password provided by Schneider Electric support). |
| 7 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

## Data Storage Password

Schneider Electric support needs a number displayed from the **Password** dialog box reached in following condition:

- In **Properties of Project** → **Protection** → **Data Storage** field, click **Reset password...** and the **Password** dialog box is displayed.

Procedure for resetting the data storage password:

| Step | Action |
|---|---|
| 1 | **Condition:** The **Password** dialog box is displayed. |
| 2 | Press `SHIFT+F2`.<br>**Result:** A grayed number is displayed in the right side of the **Password** dialog box. |
| 3 | Give this number to Schneider Electric support. |
| 4 | Receive the generated password from Schneider Electric support.<br>**Note:** The password is a temporary password, available as long as you do not modify the application. |
| 5 | Enter this password. |
| 6 | Modify the password (old password = password provided by Schneider Electric support). |
| 7 | Click **OK** or **Apply** in the **Properties of Project** window to confirm all changes.<br>If you click **Cancel** in the **Properties of Project** window, all changes are canceled. |

# Section 5.2
## Introduction to the various directories of the project browser

### About this sub-chapter

This sub-chapter describes the various directories of the project browser.

### What Is in This Section?

This section contains the following topics:

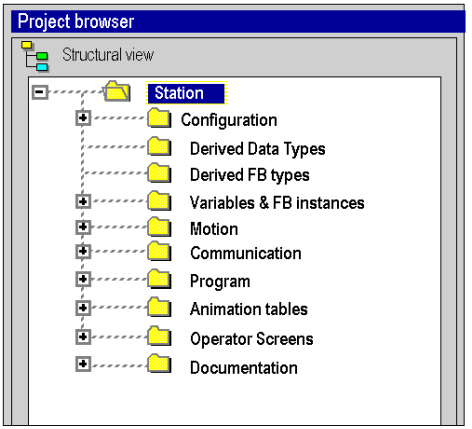

| Topic | Page |
|---|---|
| Station Directory | 194 |
| Configuration Directory | 196 |
| Derived data types (DDT) directory | 198 |
| Derived FB (DFB) types directory | 199 |
| Variables Directory | 200 |
| Motion Directory | 202 |
| Communication Directory | 204 |
| Program Directory | 206 |
| Animation Tables Directory | 209 |
| Operator Screens Directory | 211 |
| Documentation Directory | 213 |
| Conversion Report Directory | 214 |

# Station Directory

## At a Glance

The **Station** directory of the structural view allows you to access the structure of the project and the associated services.

The following illustration shows the **Station** directory:



## Associated services

The **Station** directory allows you to access the following services, which can be reached via the contextual menu:

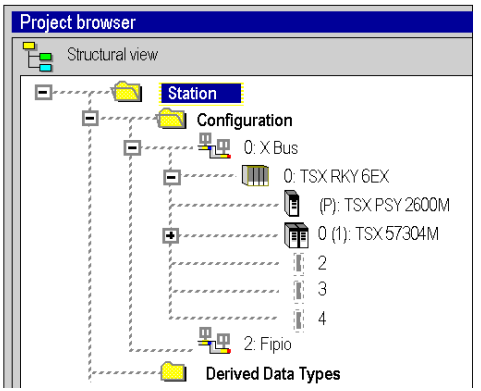
| Directory | Services |
|---|---|
| **Station** | **Export Project**: allows you to access export of the global project *(see page 1755)*.<br>**Properties**: allows you to access the properties of the global project. |
| **Configuration** | allows you to access the hardware configuration *(see page 196)* and the parameter settings of the modules. |
| **Derived Data Types** | allows you to access the DDTs *(see page 198)*. |
| **Derived FB types** | allows you to access the DFB *(see page 199)* types. |
| **Variables & FB instances** | allows you to access the variables *(see page 200)* and function block instances. |
| **Motion** | allows you to access the declaration *(see page 202)* and configuration of the servodrives. |
| **Communication** | allows you to access the configuration of the networks *(see page 204)*. |
| **Program** | allows you to access the project program *(see page 206)*. |
| **Animation tables** | allows you to access the animation tables *(see page 209)*. |
| **Operator Screens** | allows you to access the operator screens *(see page 211)*. |
| **Documentation** | allows you to access the documentation *(see page 213)*. |

# Configuration Directory

## At a Glance

The **Configuration** directory of the structural view of the project allows you to access the hardware configuration and the parameter settings of the following modules: bus, rack, module.

The following illustration shows an example of a directory tree of the **Configuration** directory:




## Accessible Services

The **Configuration** directory allows you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Configuration** | **Import**: used to import the configuration *(see page 1732)* of the project inputs/outputs (offline mode only).<br>**Import SIS**: used to import the configuration *(see page 1391)* of a project created using the SIS Automation tool.<br>**Export**: used to export the configuration *(see page 1731)* of the project inputs/outputs (offline mode only), |
| **Bus** | **Open**: used to access the bus editor, X Bus in the above example.<br>**Go to Bus Master**: displays the processor in reverse video in the project navigator, TSX 57304M in the above example. |
| **Rack** | **Open**: used to access the bus editor, TSX RKY 6EX in the above example. |
| **Module** | **Open**: used to access the input/output editor (module settings). |

## Access

From the **Configuration** directory, you can:

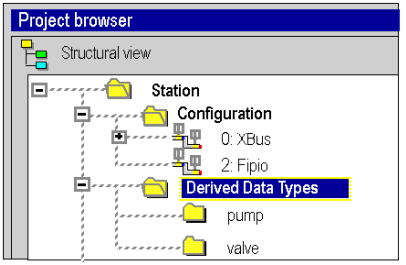
- Configure the PLC rack *(see page 1300)* with:
  - a power supply *(see page 1308)*,
  - a processor *(see page 1311)*,
  - one or more modules *(see page 1314)*.
- Configure field bus devices *(see page 1319)*,
- Access the configuration of the rack elements:
  - Premium *(see page 1363)* and Quantum *(see page 1373)* processors,
  - modules *(see page 1383)*.

# Derived data types (DDT) directory

## At a Glance

The **Derived data types** *(see Unity Pro, Program Languages and Structure, Reference Manual )* directory of the structural view of the project allows you to access the DDT types.

The following illustration shows an example of a directory tree of the **Derived data types** directory:




## Associated services

The **Derived Data Types** directory allows you to access the following services, which can be reached via the contextual menu:

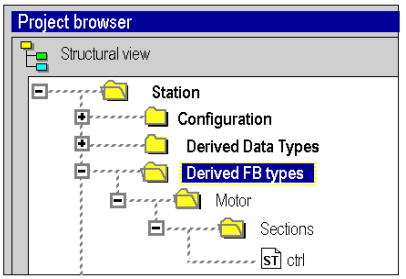
| Directory | Services |
|---|---|
| Derived Data Types | **Open**: allows you to access the **DDT types** tab of the data editor *(see page 327)*, from which you can:<br>● create *(see page 335)* a DDT,<br>● manage *(see page 346)* a DDT,<br>**Get from Library**: allows you read access to one or more DDT types from a library *(see page 293)*.<br>**Put in Library**: used to archive *(see page 355)* all the DDTs in a library *(see page 293)*.<br>**Export**: allows you to access export of all DDT types *(see page 1739)*.<br>**Import**: allows you to access import of one or more DDT types *(see page 1740)*. |
| DDT<br>(Input, Output, .etc.) | **Open**: allows you to access the DDT in the **DDT types** tab of the data editor.<br>**Delete**: allows you to delete the DDT.<br>**Put in Library**: allows you write access to the DDT in a library.<br>**Analyze**: allows you to analyze *(see page 417)* the DDT.<br>**Properties**: allows you to access the DDT properties.<br>**Export**: allows you to access export of all DDT types *(see page 1739)*. |

# Derived FB (DFB) types directory

## At a Glance

The **Derived FB types** *(see Unity Pro, Program Languages and Structure, Reference Manual )* directory of the structural view of the project allows you to access the DFB types.

The following illustration shows an example of a directory tree of the **Derived FB types** directory:




## Associated services

The **Derived FB types** directory allows you to access the following services, which can be reached via the contextual menu:

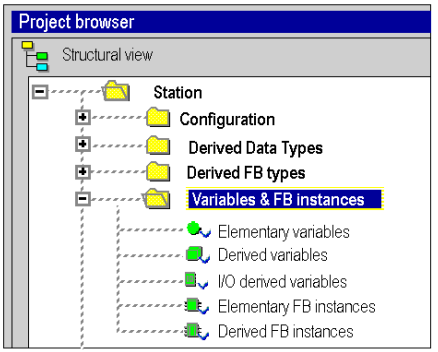
| Directory | Services |
|---|---|
| FB Type | **Open**: allows you to access the **DFB** tab of the data editor *(see page 1255)*.<br>**Get from Library**: allows you read access to one or more DFB types from a library *(see page 293)*.<br>**Put in Library**: allows you write access to all DFB types in a library.<br>**Export**: allows you to access export of all DFB types *(see page 1737)* of the project.<br>**Import**: allows you to access import of one or more DFB types *(see page 1738)*. |
| DFB Type (Counter, etc.) | **Open**: allows you to access the DFB type in the **DFB** tab of the Data Editor *(see page 1255)*.<br>**Delete**: allows you to delete the DFB type.<br>**Put in Library**: allows you write access to the DFB type in a Library *(see page 293)*.<br>**Analyze**: allows you to analyze *(see page 417)* the DFB type.<br>**Properties**: allows you to access the properties *(see page 1267)* of the DFB type.<br>**Export**: allows you to access export of the DFB type *(see page 1737)*. |
| Sections | **New Section**: allows you to create a new section *(see page 457)* in the DFB type. |
| Section XX | **Open**: allows you to access the section *(see page 457)* program language editor.<br>**Delete**: allows you to delete the section.<br>**Properties**: allows you to access the properties *(see page 461)* of the section. |

# Variables Directory

## At a Glance

The **Variables & FB instances** directory of the structural view of the project is used to access the variables (EDT, *(see Unity Pro, Program Languages and Structure, Reference Manual )*DDT, IODDT) *(see Unity Pro, Program Languages and Structure, Reference Manual )*and the function block instances(EFB, DFB). *(see Unity Pro, Program Languages and Structure, Reference Manual )*

The following illustration shows an example of the **Variables & FB instances** directory:




## Accessible Services

The **Variables & FB instances** directory allows you access to the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Variables & FB instances** | **Open**: used to access the variables editor,<br>**Export**: used to access the export of all variables *(see page 1741)* of the project,<br>**Import**: used to access the import of all variables *(see page 1744)* of the project. |
| **Elementary variables**<br>**Derived variables**<br>**Decice DDT variables**<br>**I/O derived variables**<br>**Elementary FB instances**<br>**Derived FB instances** | **Open**: used to access the tab corresponding to the variables editor,<br>**Export**: used to access the export of all variables *(see page 1741)* of the family selected (EDT, DFB, etc.) . |

## Access

From the **Variables & FB instances** directory, you can access the different tabs of the data editor:

- **Variables** tab,
  - Create *(see page 370)* a data instance,
  - Create *(see page 382)* an IODDT type data instance,
  - Create *(see page 385)* a Device DDT type data instance,
  - Modify *(see page 399)* attributes of data instances.
- **DDT Types** tab,
  - Create *(see page 335)* a data instance,
  - Archive *(see page 355)* DDTs in a library *(see page 293)*,
- **Function blocks** tab,
  - Create *(see page 360)* a data instance,
  - Modify *(see page 366)* attributes of data instances.
- **DFB Types** tab,
  - Create *(see page 1255)* a DFB type,
  - Configure settings *(see page 1256)* of a DFB type.

# Motion Directory

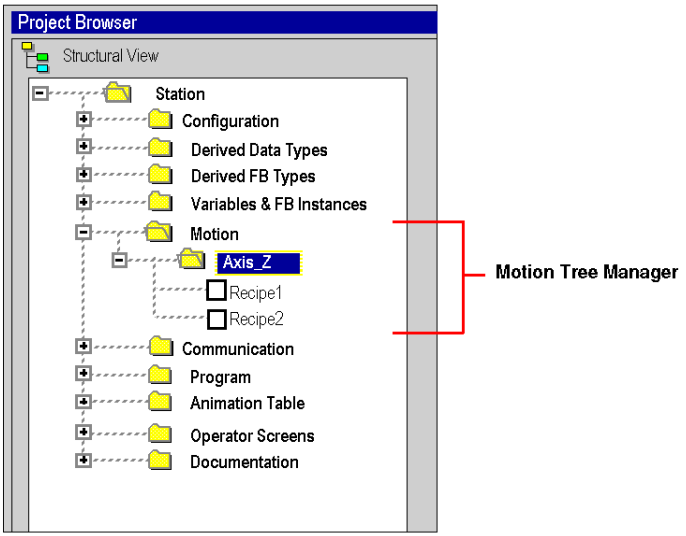
## At a Glance

The **Motion** directory of the structural view of the project allows you to access the declaration and configuration of the servodrives.

When declaring a servodrive, various information is required, such as:

- the name given to the servodrive
- the type of servodrive
- the CANopen address of the servodrive
- the reference of the servodrive
- the version of the servodrive
- the input of variable names associated to the axis.

The following diagram shows an example of a tree structure for the **Motion** directory:




In this diagram, the name given to the servodrive is ‘Axis_Z’.

A recipe is linked, by default, each time an axis is created. It is possible to create several recipes *(see MFB using Unity Pro, Start-up Guide)*.

## Accessible Services

The **Motion** directory gives you access to the following services, which can be reached via the contextual menu:

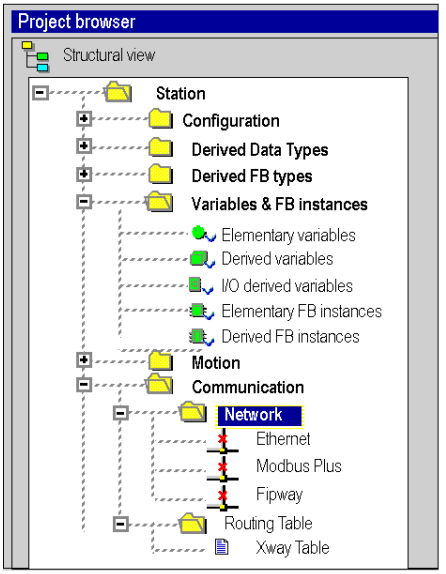
| Directory | Service |
|---|---|
| **Motion** | **New axis**: allows you to create a new axis. |
| **Axis** | **New recipe**: allows you to create a new recipe.<br>**Delete**: allows you to delete an axis.<br>**Properties**: allows you to access the axis properties. |
| **Recipe** | **Delete**: allows you to delete a recipe.<br>**Properties**: allows you to access the recipe properties. |

# Communication Directory

## At a Glance

The **Communication** directory of the structural view of the project allows you to access the configuration of the networks.

The following illustration shows an example of a directory tree of the **Communication** directory:



## Associated services

The **Communication** directory allows you to access the following services, which can be reached via the contextual menu:

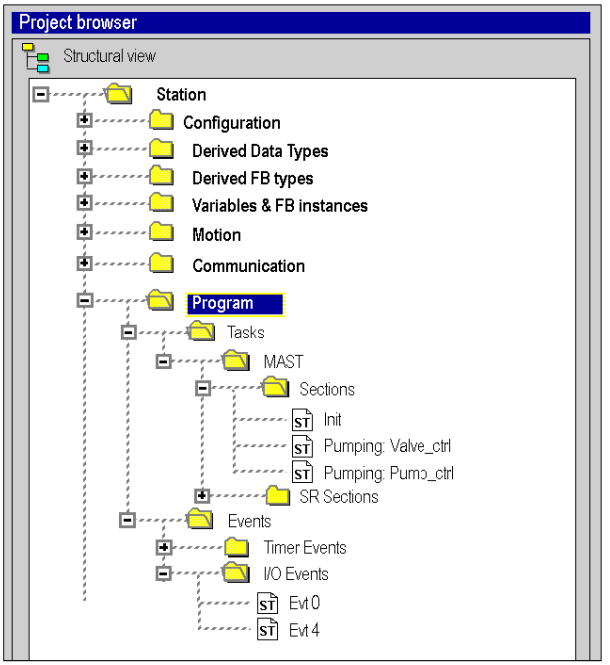
| Directory | Services |
|---|---|
| **Communication** | **Export**: allows you to access export of all networks *(see page 1745)*.<br>**Import**: allows you to access import of one or more networks *(see page 1746)*. |
| **Network** | **New Network**: allows you to add a network *(see page 427)* to your project. |
| Network (Network_1, Modbus Plus_A, etc.) | **Open**: allows you to access the corresponding network editor: Ethernet, Modbus Plus, Fipway.<br>**Export**: allows you to access export of one network *(see page 1745)*.<br>**Delete**: allows you to delete a network.<br>**Properties**: allows you to access the network properties. |
| **Routing Table** | **Open**: allows you to access the configuration *(see page 435)* of a bridge between 2 networks (routing tables). |

# Program Directory

## At a Glance

The **Program** directory of the structural view of the project allows you to define the structure of the program *(see Unity Pro, Program Languages and Structure, Reference Manual )* and to access the language editors of the program *(see Unity Pro, Program Languages and Structure, Reference Manual )* elements: sections, program modules and event processing.

The following illustration shows an example of a directory tree of the **Program** directory:



### Services associated with the sequential tasks directories.

The sequential tasks *(see Unity Pro, Program Languages and Structure, Reference Manual )* directories (Tasks, MAST, FAST, etc.) allow you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Tasks** | **New Task**: allows you to create a new sequential task *(see page 454)* (FAST, AUX, AUX0, AUX1). The MAST task is created by default.<br>**Import**: allows you to access import of sections *(see page 1735)* and/or program modules of a sequential task. |
| **MAST, FAST, etc.** | **Delete**: allows you to delete the task. The MAST task cannot be deleted.<br>**Clear**: allows you to clear the contents of the task. This clears all the sections of the task.<br>**Export**: allows you to access export of the sequential tasks *(see page 1733)*.<br>**Properties**: allows you to access the properties of the sequential task. |

### Services associated with the sections directories

The sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* directories allow you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Sections** | **New Section**: allows you to create a new "empty" section *(see page 457)*.<br>**Import**: allows you to access import of sections *(see page 1735)*.<br>**Create section activation conditions table**: allows you to initialize an animation table, with the activation condition variables associated with the sections. |
| For each section | **Open**: allows you to access the section *(see page 457)* language editor.<br>**Delete**: allows you to delete the section.<br>**Detach**: allows you to detach the section of the functional module.<br>**Export**: allows you to access export of sections *(see page 1733)*.<br>**Forcing to 0**, allows you to choose Force to 0 as the activation condition of the section (in online mode).<br>**Forcing to 1**, allows you to choose Force to 1 as the activation condition of the section (in online mode).<br>**Unforce**: allows you to choose cancel forcing as the activation condition of the section.<br>**Properties**: allows you to access the properties of the section (in online mode). |
| For each Section **Macro-step**, action, or transition | **Open**: allows you to access the section language editor. |

**NOTE:** Only the MAST task can contain one or more sections in SFC language.

## Services associated with the program modules directories

The program modules directories (SR Sections) *(see Unity Pro, Program Languages and Structure, Reference Manual )*allow you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **SR Sections** | **New SR Section**: allows you to create a new "empty" program module *(see page 481)*.<br>**Import**: allows you to access import of program module *(see page 1735)*. |
| For each program module | **Open**: allows you to access the program module language editor.<br>**Delete**: allows you to delete the program module.<br>**Export**: allows you to access export program module *(see page 1733)*.<br>**Properties**: allows you to access the properties of the program module. |

## Services associated with the event processing directories

The event processing *(see Unity Pro, Program Languages and Structure, Reference Manual )*directories allow you to access the following services, which can be reached via the contextual menu:

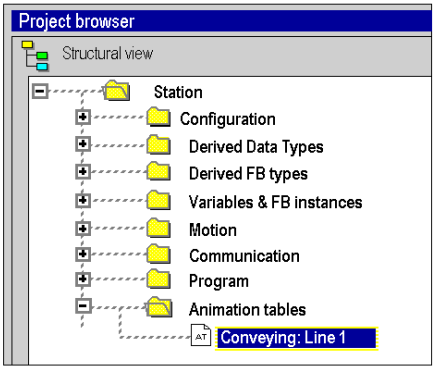
| Directory | Services |
|---|---|
| **Timer Events, I/O Events** | **New Event Section**: allows you to create a new "empty" event processing *(see page 486)*.<br>**Import**: allows you to access import of event processing *(see page 1735)*. |
| For each event processing | **Open**: allows you to access the event processing language editor.<br>**Delete**: allows you to delete the event processing.<br>**Detach**: allows you to detach the event processing of the functional module.<br>**Export**: allows you to access export of event processing *(see page 1733)*.<br>**Properties**: allows you to access the event processing properties. |

# Animation Tables Directory

## At a Glance

The **Animation Tables Directory** of the structural view of the project allows you to access the animation tables.

The following illustration shows an example of a directory tree of the **Animation Tables** directory:



## Associated services

The **Animation Tables** directory allows you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Animation tables** | **New Animation Table**: allows you to create a new animation table *(see page 1494)*.<br>**Paste**: used to paste an animation table into the clipboard or the animation tables directory.<br>**Delete all**: allows you to delete all the animation tables.<br>**Detach all**: allows you to detach all the animation tables from the functional modules.<br>**Export**: allows you to access export of all animation tables *(see page 1750)*.<br>**Import**: allows you to access import of one or more animation tables *(see page 1751)*.<br>**Make All Tables Permanent**: allows you to change all temporary animation tables to permanent animation tables (see also *Permanent and Temporary Animation Tables, page 1492*).<br>**Purge Temporary Tables**: allows you to delete all temporary animation tables (see also *Permanent and Temporary Animation Tables, page 1492*).<br>**Open Forced Bits Table**: allows you to create the animation table that includes all forced bits of a project (see also *Animation Table of Forced Bits, page 1521*). |
| For each animation table | **Open**: allows you to access the animation table.<br>**Copy**: allows you to duplicate an animation table.<br>**Paste**: allows you to add a copy of an animation table to the clipboard or the animation tables directory.<br>**Delete**: allows you to delete the animation table.<br>**Detach**: allows you to Detach the animation table from the functional module.<br>**Export**: allows you to access export of the animation table *(see page 1750)*.<br>**Properties**: allows you to access the animation table properties. |

## Access

The **Animation Tables** Directory is used to create an animation table. From this table, you can:

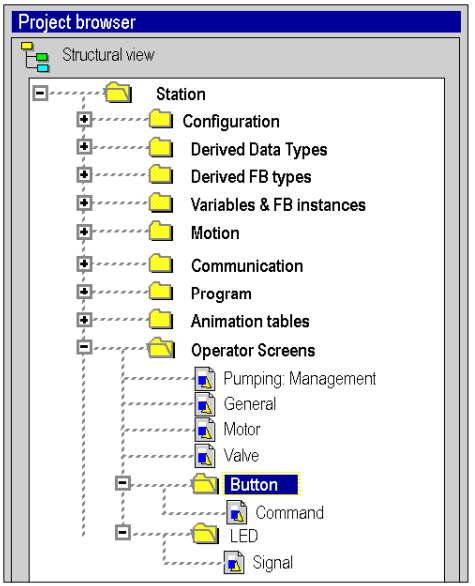
- add *(see page 1502)* data,
- switch to modification *(see page 1507)* mode,
- switch to forcing *(see page 1512)* mode,
- modify or force *(see page 1518)* several variables.

# Operator Screens Directory

## At a Glance

The **Operator Screens Directory** of the structural view of the project are used to access the operator screens.

The following diagram shows an example of a directory tree of the **Operator Screens** directory:



## Associated services

The **Operator Screens** directory allows you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Operator Screens** | **New screen**: allows you to create a new operator screen.<br>**New family**: allows you to create a new family of screens.<br>**Paste**: allows you to paste an operator screen into the clipboard or the operator screen directory.<br>**Delete all**: allows you to delete all the operator screens.<br>**Detach all**: allows you to detach all the operator screens from the functional modules.<br>**Clean unuse image**: allows you delete all the created images that are not used.<br>**Import**: allows you to access import of one or several screens *(see page 1749)* and/or family of screens.<br>**Export**: allows you to access export of all screens *(see page 1747)*.<br>**Messages list**: allows you to access the message list for the operator screens. |
| For each family of screens | **New screen**: allows you to create a new operator screen.<br>**Copy**: allows you to duplicate the family of screens and its associated operator screens.<br>**Paste**: allows you to add a copy of a screen family and its operator screens to the clipboard or the operator screens directory.<br>**Delete**: allows you to delete the family of screens.<br>**Detach**: allows you to detach the functional module family.<br>**Export**: allows you to access export of the family of screens *(see page 1747)*.<br>**Family Properties**: allows you to access the properties of the family of screens. |
| For each operator screen | **Open**: allows you to access the operator screens editor.<br>**Copy**: allows you to duplicate the operator screen.<br>**Paste**: allows you to add a copy of an operator screen to the clipboard or the operator screen directory.<br>**Delete**: allows you to delete the operator screen.<br>**Detach**: allows you to detach the operator screen from the functional module.<br>**Export**: allows you to access export of the operator screen *(see page 1747)*.<br>**Screen properties**: allows you to access the properties of the operator screen. |

## Access

The **Operator Screens** directory is used to create screens. From these screens, you can:

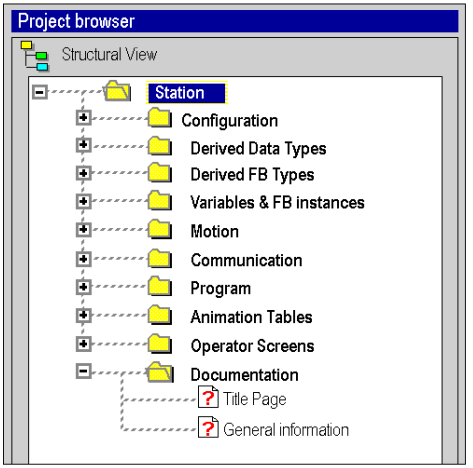
- create *(see page 1602)* objects,
- insert objects from a library *(see page 1680)*,
- modify the attributes *(see page 1616)* of the objects,
- manipulate *(see page 1645)* the objects that make up the screen,
- use the screens in online mode *(see page 1667)*.

# Documentation Directory

## At a Glance

The **Documentation** directory of the structural view of the project allows you to access the documentation.

The following illustration shows an example of a **Documentation** directory:




## Associated services

The **Documentation** directory allows you to access the following services, which can be reached via the contextual menu:

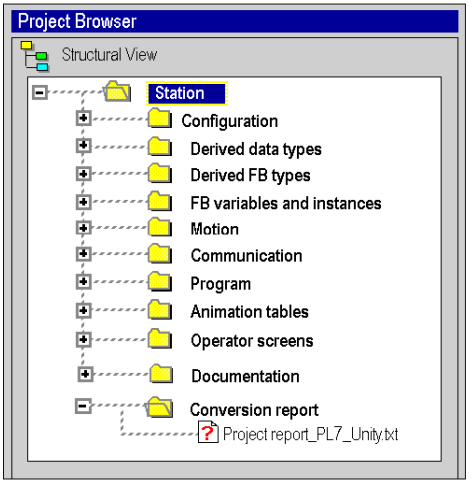
| Directory | Services |
|---|---|
| **Documentation** | **Open**: allows you to access the documentation *(see page 1695)* tool. |
| **Title Page** | **Print Configuration**: also gives access to the print configuration settings. |
| **General information** | **Open**: this heading allows you to specify various information about the project. A dialog box allows you to enter the information and to create hyperlinks *(see page 1874)*. |

# Conversion Report Directory

## At a Glance

The **Conversion Report** directory is displayed in the structural view of the project when opening a PL7 *(see Unity Pro, PL7 Application Converter, User Manual)* project with a **.FEF** extension or a Concept *(see Unity Pro, Concept Application Converter, User Manual)* project with an **.ASC** extension. This directory enables you to access a conversion report for a project.

The following illustration shows an example of a directory tree of the **Conversion Report** directory:




## Services

The **Conversion Report** directory provides you with:

- general information (application name, source file, PLC type, etc.),
- warnings,
- errors.

# Section 5.3
## Summaries of the utilities associated with the structure view and function view

## Summary of the services associated with the structural view

### At a Glance

The following tables summarize the services associated with the structural view of the project and indicate the operating modes in which you can use them.

### Station Directory

The **Station** directory *(see page 194)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Station | **Export Project**<br>**Properties** | Yes<br>Yes | No<br>Yes | No<br>Yes |

### Configuration Directory

The **Configuration** directory *(see page 196)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Configuration | **Export**<br>**Import**<br>**Import SIS** | Yes<br>Yes<br>Yes | No<br>No<br>No | No<br>No<br>No |
| Bus | **Open**<br>**Go to Bus Master** | Yes<br>Yes | Yes<br>Yes | Yes<br>Yes |
| Rack | **Open** | Yes | Yes | Yes |
| Module | **Open** | Yes | Yes | Yes |

## Derived FB Types Directory

The **Derived FB Types** directory *(see page 199)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| FB Type | **Open** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Import** | Yes | No | No |
| | **Put in Library** | Yes | No | No |
| | **Get from Library** | Yes | No | No |
| For each type of DFB | **Open** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Put in Library** | Yes | No | No |
| | **Delete** | Yes (1) | Yes (1) | Yes (1) |
| | **Analyze** | Yes | Yes | Yes |
| | **Properties** | Yes | Yes | Yes |
| Section | **Open** | Yes | Yes | Yes |
| | **Delete** | Yes | No | No |
| | **Properties** | Yes | Yes | Yes |

(1) Only if the DFB type is not instantiated in the project.

## Derived Data Types Directory

The **Derived Data Types** directory *(see page 198)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Derived Data Types | **Open** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Import** | Yes | No | No |
| | **Put in Library** | Yes | No | No |
| | **Get from Library** | Yes | No | No |
| For each DDT | **Open** | Yes | Yes | Yes |
| | **Export** | Yes | Yes | No |
| | **Put in Library** | Yes | No | No |
| | **Delete** | Yes (1) | Yes (1) | Yes (1) |
| | **Analyze** | Yes | Yes | Yes |
| | **Properties** | Yes | Yes | Yes |

(1) Only if the DDT type is not instantiated in the project.

## Variables & FB instances Directory

The **Variables & FB instances** directory *(see page 200)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Variables | **Open**<br>**Export**<br>**Import** | Yes<br>Yes<br>Yes | Yes<br>No<br>No | Yes<br>No<br>No |
| EDT, DDT, EFB, DFB | **Open**<br>**Export** | Yes<br>Yes | Yes<br>No | Yes<br>No |

## Motion Directory

The **Motion** directory *(see page 200)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Motion | **New axis** | Yes | No | No |
| Axis | **New recipe**<br>**Delete**<br>**Properties** | Yes<br>Yes<br>Yes | No<br>No<br>Yes | No<br>No<br>Yes |
| Recipe | **Delete**<br>**Properties** | Yes<br>Yes | No<br>Yes | No<br>Yes |

## Communication Directory

The **Communication** directory *(see page 204)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Communication | **Export**<br>**Import** | Yes<br>Yes | No<br>No | No<br>No |
| Network | **New Network** | Yes | No | No |
| For each network | **Open**<br>**Export**<br>**Delete**<br>**Properties** | Yes<br>Yes<br>Yes<br>Yes | Yes<br>Yes<br>Yes<br>Yes | Yes<br>Yes<br>Yes<br>Yes |
| Routing Table | **Open** | Yes | Yes | Yes |

## Program Directory

The **Program** directory *(see page 206)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Tasks | **New Task**<br>**Import** | Yes<br>Yes | No<br>No | No<br>No |
| MAST<br>FAST | **Clear**<br>**Delete**<br>**Export**<br>**Properties** | Yes<br>Yes (1)<br>Yes<br>Yes | Yes<br>No<br>No<br>Yes | No<br>No<br>No<br>Yes |
| Sections | **New Section**<br>**Create section activation conditions table**<br>**Import** | Yes<br>Yes<br>Yes | Yes<br>Yes<br>No | Yes<br>Yes<br>No |
| For each section | **Open**<br>**Delete**<br>**Detach**<br>**Export**<br>**Force to 0**<br>**Force to 1**<br>**Unforce**<br>**Properties** | Yes<br>Yes<br>Yes<br>Yes<br>No<br>No<br>No<br>Yes | Yes<br>Yes<br>Yes<br>No<br>Yes<br>Yes<br>Yes<br>Yes | Yes<br>Yes<br>Yes<br>No<br>Yes<br>Yes<br>Yes<br>Yes |
| SR Sections | **New SR Section**<br>**Import** | Yes<br>Yes | Yes<br>No | Yes<br>No |
| For each program module | **Open**<br>**Delete**<br>**Export**<br>**Properties** | Yes<br>Yes (2)<br>Yes<br>Yes | Yes<br>Yes (2)<br>No<br>Yes | Yes<br>Yes (2)<br>No<br>Yes |
| Timer Events, I/O Events | **New Event Section**<br>**Import** | Yes<br>Yes | Yes<br>No | Yes<br>No |
| For each event processing | **Open**<br>**Delete**<br>**Detach**<br>**Export**<br>**Properties** | Yes<br>Yes (2)<br>Yes<br>Yes<br>Yes | Yes<br>No<br>Yes<br>No<br>Yes | Yes<br>No<br>Yes<br>No<br>Yes |

(1) Except for MAST task.

(2) Only if the program module or the event processing is not called.

## Animation tables Directory

The **Animation tables** directory *(see page 209)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Animation tables | **New Animation Table** | Yes | Yes | Yes |
| | **Paste** | Yes | Yes | Yes |
| | **Detach All** | Yes | Yes | Yes |
| | **Delete All** | Yes | Yes | Yes |
| | **Import** | Yes | No | No |
| | **Export** | Yes | No | No |
| | **Make All Tables Permanent** | Yes | Yes | Yes |
| | **Purge Temporary Tables** | Yes | Yes | Yes |
| | **Open Forced Bits Table** | No | Yes | Yes |
| For each animation table | **Open** | Yes | Yes | Yes |
| | **Copy** | Yes | Yes | Yes |
| | **Paste** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Delete** | Yes | Yes | Yes |
| | **Detach** | Yes | Yes | Yes |
| | **Properties** | Yes | Yes | Yes |

## Operator Screens Directory

The **Operator Screens** directory *(see page 211)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Operator Screens | **New screen** | Yes | Yes | Yes |
| | **New family** | Yes | Yes | Yes |
| | **Detach All** | Yes | Yes | Yes |
| | **Delete All** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Import** | Yes | No | No |
| | **Messages list** | Yes | Yes | Yes |
| For each family of screens | **New screen** | Yes | Yes | Yes |
| | **Copy** | Yes | Yes | Yes |
| | **Paste** | Yes | Yes | Yes |
| | **Delete** | Yes | Yes | Yes |
| | **Detach** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Family Properties** | Yes | Yes | Yes |

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| For each screen | **Open** | Yes | Yes | Yes |
| | **Copy** | Yes | Yes | Yes |
| | **Paste** | Yes | Yes | Yes |
| | **Delete** | Yes | Yes | Yes |
| | **Detach** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Screen properties** | Yes | Yes | Yes |

## Documentation Directory

The **Documentation** directory *(see page 213)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Documentation | **Open** | Yes | Yes | Yes |
| Title Page | **Print Setup** | Yes | Yes | Yes |
| General Information | **Open** | Yes | Yes | Yes |

# Chapter 6
# Functional modules

## Aim of this Chapter

This chapter introduces the functional modules incorporated in a Unity Pro project.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 6.1 | Introduction to the functional view | 222 |
| 6.2 | The functional modules and their associated utilities | 234 |

# Section 6.1
## Introduction to the functional view

### About this sub-chapter

This sub-chapter provides general information on the functional modules and their associated utilities.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Functional modules | 223 |
| Station directory of the functional view | 225 |
| Functional Module Directory | 226 |
| Functional Module Program Directory | 228 |
| Functional Module Animation Tables Directory | 229 |
| Functional Module Operator Screens Directory | 230 |
| Summary of the services associated with the functional view | 231 |

# Functional modules

## Definition

A functional module is a group of program elements intended to perform a PLC function.

A functional module is comprised:
- program or event processing sections,
- animation tables associated with the functional module,
- runtime screens associated with the functional module,
- a comment (max. 1024 characters),
- interleaved functional modules, with these modules, in relation to the main function, performing one or more PLC sub-functions.

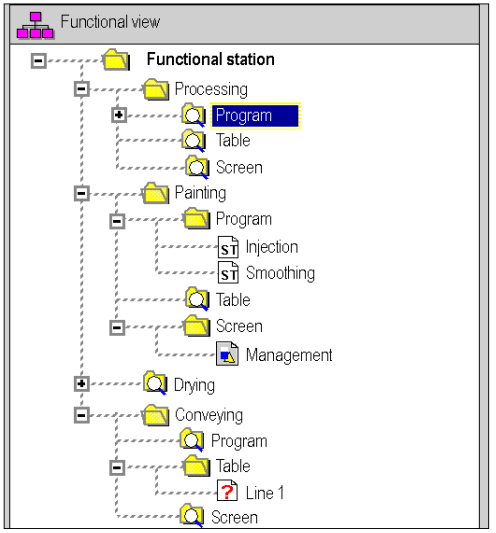
**NOTE:** a functional module does not necessarily have a program section, an animation table or runtime screens.

The other elements that make up a project are not incorporated into the functional modules:
- the configuration,
- the global variables of a project,
- the DFB and DDT types, which are global to a project,
- the SRi program modules, which are global to a task,
- the documentation.

## At a glance

All the functional modules of the project can be found in the **Functional Station** directory of the function view.

The following illustration shows the **Functional Station** directory:




As the default the browser displays the first level of the directory tree. To access the other levels, you need to expand the directories.

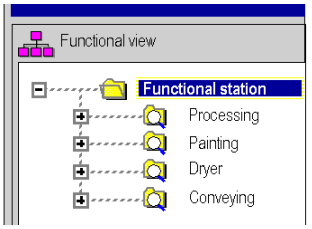
**NOTE:** This breakdown does not take into account the order of execution of the program by the PLC.

# Station directory of the functional view

## At a Glance

The **Functional Station** directory of the functional view allows you to access all the functional modules of the project and the associated services.

The following illustration shows the **Functional Station** directory:




## Associated services

The **Functional Station** directory allows you to access the following services, which can be reached via the contextual menu:

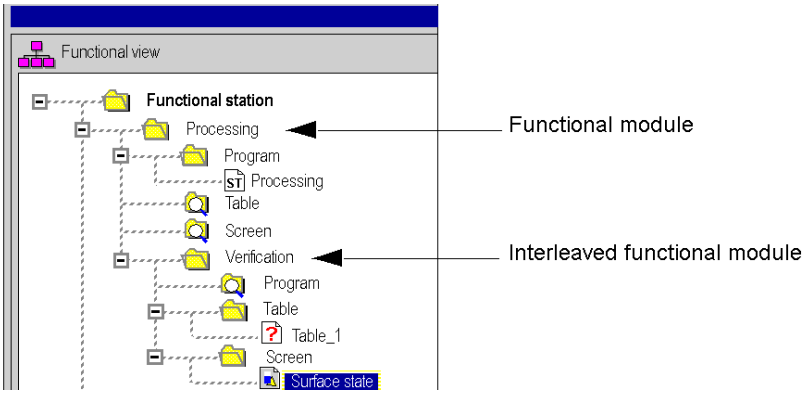
| Directory | Services |
|---|---|
| **Functional station** | **New Functional Module**: allows you to create a new functional module.<br>**Detach all**: allows you to detach all elements included in the functional modules (sections and animation tables) without deleting them.<br>**Export**: allows you to access export of the global project *(see page 1755)*.<br>**Import**: allows you to access import of a functional module *(see page 1754)*.<br>**Properties**: allows you to access the properties of the global project. |
| **For each functional module** | allows you to access all the elements of the functional module: program, animation tables, operator screens, nested functional modules. |

# Functional Module Directory

## At a Glance

The **Functional Module** directory of the functional view allows you to access all the elements of a functional module (program, animation tables and operator screens) and the nested modules.

The following illustration shows an example of a **Functional Module** directory:



## Associated services

The **functional module** directory allows you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Functional module: Factory1** | **New Functional Module**: allows you to create a new nested functional module.<br>**Create**: allows you to create a new program element in the functional module: section, program module or event processing.<br>**Delete**: allows you to delete the functional module.<br>**Detach all**: allows you to detach all elements included in the functional module (sections and animation tables) without deleting them.<br>**Export**: allows you to access export of the functional module *(see page 1752)*.<br>**Import**: allows you to access import of a functional module *(see page 1754)*.<br>**Create section activation conditions table/CTRL+T**: allows you to initialize an animation table, with the activation conditions associated with the program elements of the functional module (sections, program modules and event processings).<br>**Protection of Included Sections**: allows you to define the protection of the program elements (sections, program modules and event processings) of the function module (no protection, write-protection, read/write-protection).<br>**Properties**: allows you to access the functional module properties. |
| **Program** | allows you to access the functional module programs *(see page 228)*. |
| **Table** | allows you to access the animation tables of the functional module *(see page 229)*. |
| **Screen** | allows you to access the operator screens of the functional module *(see page 230)*. |

# Functional Module Program Directory

## At a Glance

The **Program** Directory of a functional module allows you to access the program elements of this module: sections, *(see Unity Pro, Program Languages and Structure, Reference Manual )*program modules and event processing. *(see Unity Pro, Program Languages and Structure, Reference Manual )*

## Associated services

The **Program** directory of a functional module allows you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Program** | **Create**: allows you to create a new program element in the functional module: section, program module or event processing.<br>**Detach all**: allows you to detach all elements included in the functional module (sections and animation tables) without deleting them.<br>**Import**: allows you to access import program elements of the functional module: sections, program modules, event processing (see also topic on program import *(see page 1735)*).<br>**Create section activation conditions table/CTRL+T**: allows you to initialize an animation table, with the activation conditions associated with the program elements of the functional module: sections.<br>**Protection of Included Sections**: allows you to define the protection of the program elements of the functional module: sections, program modules and event processing. |
| **Section, SR section, event processing** | See *Program Directory, page 206* for a description of the associated services. |

# Functional Module Animation Tables Directory

## At a Glance

The **Table** directory of a functional module allows you to access the animation tables of this module.

## Associated services

The **Table** directory of a functional module allows you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Table** | **New Animation Table**: allows you to create a new animation table.<br>**Detach all**: allows you to detach all the animation tables from the functional module, without deleting them.<br>**Import**: allows you to access import of the animation tables *(see page 1751)* in the functional module. |
| **For each animation table** | **Open**: allows you to access the animation table.<br>**Copy**: allows you to duplicate the animation table.<br>**Paste**: allows you to add a copy of an animation table into the animation tables directory in the functional module and the project browser.<br>**Export**: allows you to access export of the animation table *(see page 1750)*.<br>**Delete**: allows you to delete the animation table.<br>**Detach**: allows you to detach the animation table from the functional module, without deleting it.<br>**Properties**: allows you to access the animation table properties. |

# Functional Module Operator Screens Directory

## At a Glance

The **Screen** directory of a functional module allows you to access the operator screens of this module.

## Associated services

The **Screen** directory of a functional module allows you to access the following services, which can be reached via the contextual menu:

| Directory | Services |
|---|---|
| **Screen** | **Import**: allows you to access import of one or more screens *(see page 1749)* and/or families of screens.<br>**New family**: allows you to create a new family of screens.<br>**New screen**: allows you to create a new operator screen.<br>**Detach all**: allows you to detach all the screen elements from the functional module, without deleting them. |
| **For each family of screens** | See *Operator Screens Directory, page 211* for a description of the screen families services. |
| **For each operator screen** | See *Operator Screens Directory, page 211* for a description of the services available from the screen. |

# Summary of the services associated with the functional view

## At a Glance

The following tables summarize the services associated with the functional view of the project and indicate the operating modes in which you can use them.

## Functional Station Directory

The **Functional Station** directory *(see page 225)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Functional Station | **New Functional Module** | Yes | Yes | Yes |
| | **Detach All** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Import** | Yes | No | No |
| | **Properties** | Yes | Yes | Yes |

## Functional Module Directory

The **Functional Module** directory *(see page 226)* allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| For each functional module | **New Functional Module** | Yes | Yes | Yes |
| | **Detach All** | Yes | Yes | Yes |
| | **Create** | Yes | Yes | Yes |
| | **Delete** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Import** | Yes | No | No |
| | **Protection of Included Sections** | Yes | Yes | Yes |
| | **Create section activation conditions table** | Yes | Yes | Yes |
| | **Properties** | Yes | Yes | Yes |

## Functional Module Program Directory

The **Program** directory *(see page 228)* of a functional module allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Program | **Create** | Yes | Yes | Yes |
| | **Import** | Yes | No | No |
| | **Detach All** | Yes | Yes | Yes |
| | **Protection of Included Sections** | Yes | Yes | Yes |
| | **Create section activation conditions table** | Yes | Yes | Yes |
| For each section, SR sections, event processing | **Open** | Yes | Yes | Yes |
| | **Detach** | Yes | Yes | Yes |
| | **Delete** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Force to 0** | No | Yes | Yes |
| | **Force to 1** | No | Yes | Yes |
| | **Unforce** | No | Yes | Yes |
| | **Properties** | Yes | Yes | Yes |

## Functional Module Animation Tables Directory

The **Table** directory *(see page 229)* of a functional module allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Table | **New Animation Table** | Yes | Yes | Yes |
| | **Import** | Yes | No | No |
| | **Detach All** | Yes | Yes | Yes |
| For each animation table | **Open** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Delete** | Yes | Yes | Yes |
| | **Copy** | Yes | Yes | Yes |
| | **Paste** | Yes | Yes | Yes |
| | **Properties** | Yes | Yes | Yes |
| | **Detach** | Yes | Yes | Yes |

## Functional Module Operator Screens Directory

The **Screen** directory *(see page 230)* of a functional module allows you to access the following services.

| Directory | Services | Mode | | |
|---|---|---|---|---|
| | | Off-line | On-line Stop | On-line Run |
| Screen | **New family** | Yes | Yes | Yes |
| | **New screen** | Yes | No | No |
| | **Import** | Yes | No | No |
| | **Detach All** | Yes | Yes | Yes |
| For each family of screens | **New screen** | Yes | Yes | Yes |
| | **Copy** | Yes | Yes | Yes |
| | **Paste** | Yes | Yes | Yes |
| | **Delete** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Detach** | Yes | Yes | Yes |
| | **Family Properties** | Yes | Yes | Yes |
| For each operator screen | **Open** | Yes | Yes | Yes |
| | **Copy** | Yes | Yes | Yes |
| | **Paste** | Yes | Yes | Yes |
| | **Export** | Yes | No | No |
| | **Detach** | Yes | Yes | Yes |
| | **Delete** | Yes | Yes | Yes |
| | **Screen properties** | Yes | Yes | Yes |

# Section 6.2
## The functional modules and their associated utilities

### About this sub-chapter

This sub-chapter is a guide on how to use the utilities associated with the functional modules.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Properties of a functional module | 235 |
| Creating a functional module | 236 |
| Programming a functional module | 237 |
| Protecting program elements of the functional module | 239 |
| Debugging a functional module | 240 |
| Detaching/Deleting a functional module | 241 |
| Functional module export | 245 |
| Functional module import | 246 |
| Creating, Deleting, Locating, Dragging-and-Dropping an Animation Table in a Functional Module | 247 |
| Creating, Deleting, Locating and Dragging-and-Dropping an Operator Screen in a Functional Module | 250 |

# Properties of a functional module

## Properties to define

List of different properties to define:

- the name comprising 32 characters. This name must be unique for modules positioned on the same level.
- the definition of the activation conditions of the program elements (sections, program modules and event processings) of the functional module (Force to 0, Force to 1, cancel forcing),
- the protection of the program elements (sections, program modules and event processings) of the functional module (no protection, write-protection, read/write-protection),
- the comment comprising 256 characters.

## Displaying or editing properties

Carry out the following steps:

| Step | Action |
|---|---|
| 1 | In the functional view, right-click the functional module, in the **Functional Station** directory. |
| 2 | Click **Properties**. |
| 3 | Perform the editing. |
| 4 | Confirm with **OK**. |

**NOTE:** The **Apply** button confirms the editing without closing the window.

# Creating a functional module

## At a Glance

A functional module can be created off-line, with the PLC at Stop or in Run.

It can be created at the Station level or at the level of each existing functional module.

## Creating a new functional module

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the functional view, right-click the **Functional Station** directory. |
| 2 | Select **New Functional Module**. |
| 3 | Enter the name and comment and confirm with **OK**. |

## Creating a lower level functional module

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the functional view, right-click the module "above", in the **Functional Station** directory. |
| 2 | Select **New Functional Module**. |
| 3 | Enter the name and comment and confirm with **OK**. |

## Moving a functional module

A functional module can be moved off-line, with the PLC at **Stop** or in **Run** (this has no effect on the execution of the project). The movement corresponds only to a modification of the functional architecture of the project (a module is directly attached to the **Functional Station** directory level or to another functional module).

Moving a functional module:

| Step | Action |
|---|---|
| 1 | Left-click on the module to be moved (holding down the mouse button), |
| 2 | Move the module to the desired position. |

# Programming a functional module

## Introduction to programming a functional module

A functional module has a **program** directory that may contain:

- LD, ST, IL sections *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- event processing *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- SFC sections *(see Unity Pro, Program Languages and Structure, Reference Manual )*

Various cases may arise when programming a functional module:

- **Case 1**: the section, event processing or SFC section already exists in the structural view
- **Case 2**: creating the section or event processing using the functional view
- **Case 3**: importing a section using the functional view

## Case 1: the section, event processing or chart already exists in the structural view

The section has already been created in the structural view:

| Step | Action |
|---|---|
| 1 | Select the section. |
| 2 | Move the section to the functional module. |

## Case 2: creating the section or event processing using the functional view

The following table shows how to create a section:

| Step | Action |
|---|---|
| 1 | In the **Functional view**, right-click the **Program** directory in the **Functional Station** directory. |
| 2 | Select the **Create** command from the contextual menu. |
| 3 | Select either **New Section** or **New Event**. |
| 4 | Enter the various headings in exactly the same way as for creating a section from the structural view. The name of the functional module is shown again at the structural view level. |

### Case 3: importing a section using the functional view

The following table shows how to import a section:

| Step | Action |
|---|---|
| 1 | In the **Functional view**, right-click the **Program** directory in the **Functional Station** directory. |
| 2 | Select the **Import** command from the contextual menu. |
| 3 | In the **Import dialog** select the file to be imported. |
| 4 | Click on the **Import button** to finish the import. |

When using the functional view to import a section that does not have its task in the importing application, the section is attached to the **MAST** task.

**⚠ WARNING**

**UNEXPECTED APPLICATION BEHAVIOR**

Verify that an imported section operates correctly while attached to the **MAST** task if it was originally running in a non-MAST task.

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

### Rules

Take into account the following notes:

| | |
|---|---|
| 1 | An SFC section can only be created off-line and then only in the **Mast** task. |
| 2 | A macro-step or an event can only be created off-line. |
| 3 | The other actions are authorized in off-line mode, with the PLC at **Stop** or in **Run**. |
| 4 | The protection of a module applies to all the sections attached to the functional module and to the lower level modules. |

# Protecting program elements of the functional module

## At a Glance

In a functional module you are able to define the protection *(see page 186)* of the program elements (sections, program modules and event processings) contained in the module.

## Protection

Procedure:

| Select from | Then... |
|---|---|
| the functional module | ● with the mouse right-click the **Program** directory in a functional module of the functional view,<br>● select **Protection of Included Sections**,<br>● select **Write** or **Read/Write**.<br>**Result**: All the program elements of the module and of the lower module(s) are protected. |
| from the **Program** directory | ● with the mouse right-click on the **Program** directory in a functional module of the functional view,<br>● select **Protection of Included Sections**,<br>● select **Write** or **Read/Write**.<br>**Result**: All the program elements of the module are protected. |

# Debugging a functional module

## At a Glance

The organization of a functional module, and the distribution of the sections, event processing and SFC sections in the various modules has no impact on the execution of the program. The program is executed in the order shown in the structural view. To help you debug a functional view, the following are available:

- basic debugging functions,
- the animation table initialization function.

## Initializing an animation table

This action allows you to initialize an animation table, with the activation conditions associated with the program elements of the functional module (sections, program modules and event processing).

| Select from | Then... |
|---|---|
| the functional module | • right-click with the mouse on the directory of a functional module in the functional view.<br>• select **Create section activation conditions table**. |
| from the **Program** directory | • with the mouse right-click on the **Program** directory in a functional module of the functional view.<br>• select **Create section activation conditions table**. |

**NOTE:** All the activation conditions that are displayed in the animation table are theconditions used during the last build (partial or total).

# Detaching/Deleting a functional module

## At a Glance

Detaching elements from a functional module consists of cutting the links between a functional module and the associated elements (program elements, animation table and screen).

- the sections contained in the module are not deleted, they are merely detached,
- the animation tables associated with the modules are not deleted, they are merely detached from the module,
- the screens associated with the modules are not deleted, they are merely detached from the module.

**NOTE:** These actions are authorized off-line and on-line, with PLC at Stop or in Run.

## Detaching Functional Modules

To detach functional modules carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the **Functional Station** directory. |
| 2 | Select **Detach All** from the contextual menu. |
| 3 | Confirm with **Yes**. |

## Detaching a functional module

To detach a functional module carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the functional module from the **Functional Station** directory. |
| 2 | Select **Detach All** from the contextual menu. |
| 3 | Confirm with **Yes**. |

## Detaching all the elements in a directory from a functional module

To detach all the elements in a directory (Program, Table, Screen) from a functional module, you must do the following:

| Step | Action |
|---|---|
| 1 | Select the **Program** or **Table** or **Screen** directory from the functional module. |
| 2 | Select the **Detach All** command from the contextual menu.<br>**Result**: All the elements in the chosen directory are detached from the functional module. |

### Detaching one element in a directory from a functional module

To detach one element in a directory (Program, Table, Screen) from a functional module, you must do the following:

| Step | Action |
|---|---|
| 1 | Select the element to be detached in the **Program** or **Table** or **Screen** directory. |
| 2 | Select**Detach** from the contextual menu.<br>**Result**: The element in the chosen directory is detached from the functional module. |

### Introduction to deleting a functional module

Deletion can be carried out in a number of ways:

- deleting one or more functional modules without deleting the program elements, animation tables and screens,
- deleting one functional module with deletion of the program elements, animation tables and screens,
- deleting all the functional modules with deletion of the program elements, animation tables and screens,
- deleting one section, a macro-step or an event processing in a functional module.

### Deleting one or more functional modules without deleting the program elements, animation tables and screens

To delete a functional module without deleting the program elements and animation tables, you must proceed as follows:

| Step | Action |
|---|---|
| 1 | From the functional module, detach the module with the **Detach all** command in the contextual menu.<br>**Result**: All the elements of the module and the lower level modules are detached. |
| 2 | Delete the module(s) by right-clicking (contextual menu) and selecting the **Delete** command.<br>**Note:** You must be in **off-line** mode. |

### Deleting one functional module with deletion of the modules, animation tables and screens

**⚠ CAUTION**

**UNEXPECTED APPLICATION BEHAVIOR - MODULE DELETION**

Before deleting a module, ensure that the deletion of the subsequent elements:
- the sub-modules contained in the module,
- the sections contained in the module,
- the associated animation tables,
- the associated screens,

has no adverse consequence on the application behavior.

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

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the module. |
| 2 | Select **Delete**. |
| 3 | Confirm with **Yes**. |

**NOTE:** This action is authorized for the SFC sections and the events in off-line mode, with the PLC at **Stop**. The other elements may be deleted in **Run**.

### Deleting a section or an event in a functional module

**⚠ CAUTION**

**UNEXPECTED APPLICATION BEHAVIOR - SECTION DELETION**

A section or an event may be deleted in a functional module. They are then deleted in the module and in the associated task.

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

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the desired section or event. |
| 2 | Select **Delete**. |
| 3 | Confirm with **Yes**. |

## Rules

Take into account the following rules:

| | |
|---|---|
| 1 | In order not to delete the contents of a module, you must detach its elements before deleting. |
| 2 | An SFC section or an event can only be deleted in off-line mode. |

# Functional module export

## Exporting a functional module

You can export *(see page 1752)* a functional module.

# Functional module import

## Importing a functional module

You can import *(see page 1754)* a functional module.

# Creating, Deleting, Locating, Dragging-and-Dropping an Animation Table in a Functional Module

## At a Glance

In a functional module you are able to create, delete, locate and drag-and-drop an animation table.

## Creating an Animation Table

Procedure:

| If the table | Then... |
|---|---|
| already exists | • select the animation table in the **Functional Station** directory of the functional view or the **Animation tables** directory of the structural view,<br>• drop the animation table to a functional module in the **Table** directory level. |
| must be created from the functional view | • with the mouse right-click the **Table** directory of the module,<br>• select **New Animation Table**. |

## Accessing the Properties of an Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the table. |
| 2 | Select **Properties** from the contextual menu. |
| 3 | You can edit the name or the comment of a table and its assignment to a module. |

## Copying an Animation Table.

Procedure:

| If the table | Then... |
|---|---|
| must be copied from the functional view. | • with the mouse right-click the source table in the **Functional Station** directory of the functional view,<br>• select **Copy**,<br>• select the source table in the **Functional Station** directory of the functional view,<br>• select **Paste**.<br>**Result**: The copy of the table is created in the source table directory. |

## Opening an Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the table. |
| 2 | Select **Open** from the contextual menu. |

## Deleting an Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the table. |
| 2 | Select **Delete** from the contextual menu. |

## Moving an Animation Table

Procedure:

| If using | Then... |
|---|---|
| the drag-and-drop function | • left-click on the table to be moved (holding down the mouse button),<br>• drop the table at the desired position.<br>Note: The insertion point of the table is represented by a line. |
| **Properties** from the contextual menu | • select the table,<br>• select **Properties** from the contextual menu,<br>• select the name of the module in the **Functional Module** zone. |

## Detach all Animation Tables

Procedure:

| If using the contextual menu | Then... |
|---|---|
| Detach all | • select the **Table** directory of the functional module,<br>• select **Detach All** from the contextual menu,<br>• confirm with **Yes**. |

## Detaching an Animation Table

Procedure:

| If using the contextual menu | Then... |
|---|---|
| Detach | ● select the animation table,<br>● select **Detach** from the contextual menu,<br>● confirm with **Yes**. |
| **Properties** | ● select the table,<br>● select **Properties** from the contextual menu,<br>● select "none" in the **Functional Module** zone. |

# Creating, Deleting, Locating and Dragging-and-Dropping an Operator Screen in a Functional Module

## At a Glance

In a functional module you are able to create, delete, locate and drag-and-drop an operator screen.

## Creating an Operator Screen

Procedure:

| If the screen | Then... |
|---|---|
| already exists | ● select the operator screen in the **Functional Station** directory of the functional view or the **Operator Screens** directory of the structural view,<br>● in any language editor window, select **Open Operator Screen.** in the function block contextual menu.<br>● drop the operator screen in a functional module at the **Screen** directory level.<br>**NOTE:** With the **Open Operator Screen** menu if multiple operator screens exist with the same name, the editor opens the first one found. |
| must be created from the functional view | ● with the mouse right-click the **Screen** directory of the functional module,<br>● select **New screen**. |

## Accessing the Operator Screen Properties

Procedure:

| If selected at the level of the | Then... |
|---|---|
| a family of screens | ● select the family of screens in the **Screen** directory of the structural view,<br>● Select **Family Properties** from the contextual menu. |
| a specific screen | ● select a screen in the **Screen** directory of the structural view,<br>● Select **Screen Properties** from the contextual menu. |

## Creating a Family of Operator Screens

Procedure:

| If the family | Then... |
|---|---|
| already exists | • select the family of operator screens in the **Functional Station** directory of the functional view or the **Operator Screen** directory of the structural view,<br>• drop the family of operator screens in a functional module at the **Screen** directory level. |
| must be created from the functional view | • with the mouse right-click the **Screen** directory of the functional module,<br>• select **New family**. |

## Opening an Operator Screen

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the operator screen in the **Functional Station** directory of the functional view or the **Operator Screens** directory of the structural view. |
| 2 | Select **Open** from the contextual menu. |

## Deleting an Operator Screen

Procedure:

| If the screen | Then... |
|---|---|
| does not belong to a family of screens | • with the mouse right click the operator screen in the **Operator Screen** directory,<br>• select **Delete**. |
| belongs to a family of screens | • from the **Screen** directory, right-click the operator screen belonging to the family,<br>• select **Delete**. |

## Moving an Operator Screen

Procedure:

| If using | Then... |
|---|---|
| the drag-and-drop function | • left-click on the operator screen to be moved (holding down the mouse button),<br>• move the operator screen to the desired location.<br>**NOTE:** The insertion point of the operator screen is represented by a line. |
| **Screen Properties** from the contextual menu | • select the operator screen,<br>• select **Screen Properties** from the contextual menu,<br>• select the name of the module in the **Location** zone. |

## Detaching All Screens and Families of Screens

Procedure:

| If you have chosen to detach | Then... |
|---|---|
| the operator screens and the families of screens | • select the **Screen** directory of the functional module,<br>• select **Detach All** from the contextual menu,<br>• confirm with **Yes**. |
| the family of screens | • in the **Screen** directory of the functional module, select the family of screens,<br>• select **Detach** from the contextual menu, |
| a specific screen | • select the operator screen in the **Screen** directory of the functional module,<br>• select **Detach** from the contextual menu, |

## Accessing the Messages List

The following table shows the procedure for accessing the operator screens messages list.

| Step | Action |
|---|---|
| 1 | Select the **Operator Screens** directory from the structural view. |
| 2 | Select **Messages list** from the contextual menu. |

# Chapter 7
# FDT Container

## Object of the Chapter

This chapter introduces the FDT Container and explains how it is used.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 7.1 | FDT Container | 254 |
| 7.2 | DTM Hardware Catalog | 257 |
| 7.3 | DTM Browser | 262 |
| 7.4 | DTM Properties Tabs | 288 |

# Section 7.1
# FDT Container

## Object of this Section

This section is an introduction to using DTMs in Unity Pro.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction to the Unity Pro FDT Container | 255 |
| FDT Container User Rights | 256 |

# Introduction to the Unity Pro FDT Container

## At a Glance

The FDT/DTM (**Field Device Tool**/**Device Type Manager**), an open-standard, in Unity Pro integrates the configuration tools needed to build projects, with a mix of Schneider and Third Party intelligent devices, on heterogeneous networks.

Unity Pro is now a **FDT Container** (Field Device Tool Container) for DTMs and, therefore, is a Generic Fieldbus Intergrater.

**NOTE:** Schneider-Electric is not responsible for the correct functioning of non-Schneider-Electric DTMs under Unity Pro.

## FDT Concept

The FDT defines the interfaces between device-specific software components (DTMs) and Unity Pro. The focus of the current version of FDT is on engineering, commissioning, diagnostics and documentation of fieldbus based control systems.

The device-specific software component is called a DTM (Device Type Manager), which are supplied by the device manufacturer together with the device. Because the FDT technology used in Unity Pro is based on an open standard, Unity Pro accepts compliant DTM products from any DTM/device vendor.

**NOTE:** The DTMs are installed/uninstalled on the PC (like most other Windows® programs) that runs Unity Pro and information about the DTMs is kept in the Windows Registry.

**NOTE:** The DTM Browser displays the DTM connectivity configuration, but does not create any topological variables (%I or %Q) in the application.

## Components

To create a Fieldbus network with Unity Pro, its FDT Container uses:

- The DTM Hardware Catalog *(see page 257)*
- The DTM Browser *(see page 262)*
- The DTM Editor, which uses vendor supplied GUIs to configure the DTMs
  **NOTE:** The DTM vendor is responsible for the DTM Configuration on-line Help, it is not available through the Unity Pro on-line Help.

# FDT Container User Rights

## At a Glance

With 2 exceptions, a **FDT Container** user has unlimited access to the installed DTMs.

## User Rights

The **FDT Container** user has all rights of access for the DTM connectivity tree (add, delete, configure, save DTMs) unless:

1. The Unity Pro profile is set to Read Only for the user
2. The user has opened Unity Pro in the Read Only mode

## Unity Pro Online/Offline and Device Connections

For the user to load and store DTM device parameters from/to its physical device, the device must be connected to its DTM through the Host PC.

This connection:

- Requires that the DTM be installed on the Host PC.
- Is **independent** of the Unity Pro - PLC connection status. A DTM can connect to its device and transfer configuration information over the network in either direction under both conditions: Unity Pro off-line or Unity Pro on-line.

## Limitation

If the user opens a project in Read Only mode, the DTM user right remains in Read-Only as long as the project is still opened (even if the user removes the Read-Only opened mode by performing the service Save As).

# Section 7.2
# DTM Hardware Catalog

## Description of DTM Hardware Catalog

### At a Glance

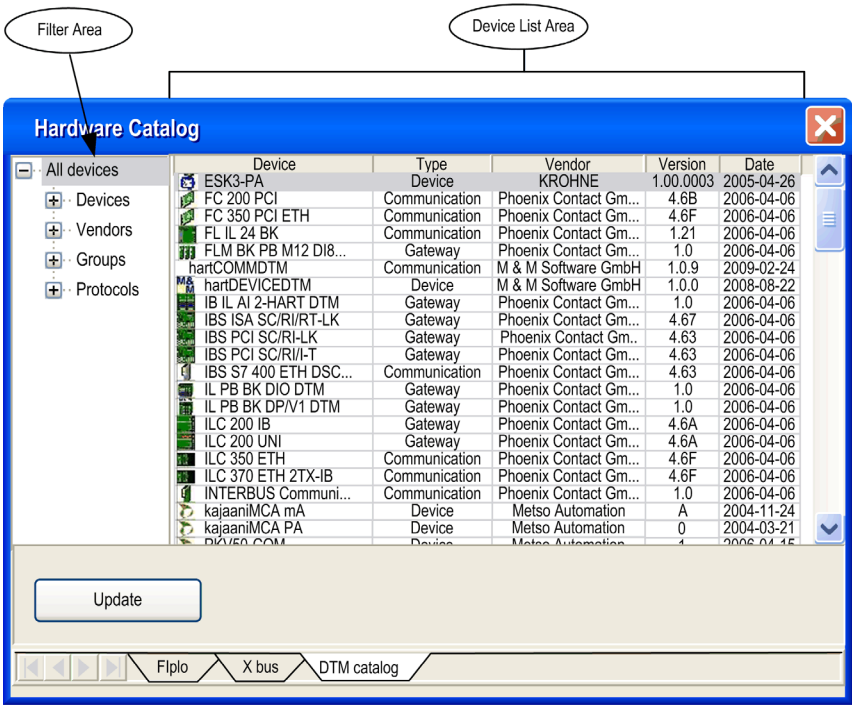
The DTM **Hardware Catalog** displays all the DTMs installed on the PC.

The DTM tab of the **Hardware Catalog** is divided into 3 parts:

- The Filter area
- The Device List Area
- The Update button

### The DTM Catalog Tab

The following illustration is an example of the DTM **Hardware Catalog**:



### Add DTMs to Connectivity Tree

There are 2 ways to add DTMs to a DTM connectivity tree:

- By using the Add device *(see page 277)* procedure in the DTM Browser
- By using the mouse to drag a DTM from the Hardware Catalog and dropping it on an appropriate DTM in the connectivity tree

### The Filter Area

The browser in the **Filter Area** has 4 types of filters with sub-filters. The actual sub-filters available depend on the type of DTMs installed on the PC:

- Devices
  There are 3 device type sub-filters:
  - Communication
  - Gateway or Modular
  - (Other) Devices
- Vendors
  Selecting a vendor name displays only the installed DTMs from that vendor.
- Groups
  Selecting a group type displays only the installed DTMs corresponding to that type of DTM.
- Protocols
  Selecting a protocol displays only the installed DTMs corresponding to that type of protocol. When a protocol is selected as the filter, a DTM is only shown if this protocol is required and/or supported by the DTM.

When a filter is selected, only the DTMs corresponding to that filter are displayed in the **Device List Area**.

### The Device List Area

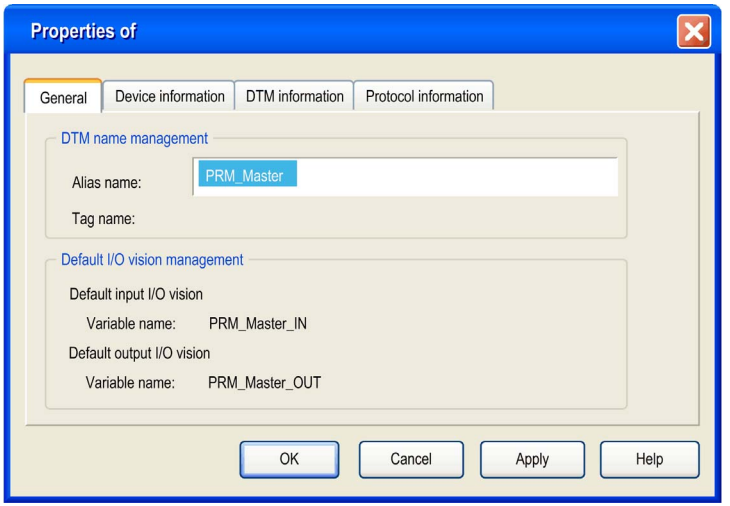
The **Device List Area** columns give information about each DTM. They are:

- Device (name)
- Type
- Vendor (name)
- Version
- Date

Clicking on a column heading sorts the DTMs alphabetically according the contents of the column.

## DTM Properties

To display the properties of a DTM from the **Device List Area**:

| Step | Action |
|---|---|
| 1 | Click anywhere on the line of the DTM.<br>Result: The DTM is selected. |
| 2 | Right-click on the DTM.<br>Result: The Properties menu opens. |
| 3 | Use the tabs *(see page 288)* to see additional information about the DTM:<br>  |
| 4 | When finished with the window, click on **OK** or **Cancel**. |

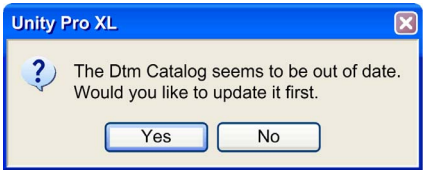
**NOTE:** When installing or modifying a DTM, update the Hardware Catalog *(see page 260)* to update its properties information.

## Update Hardware Catalog

The DTM catalog is fully embeded in the Unity Pro software and requires to be up to date whenever a DTM is installed or unistalled. It can be done:

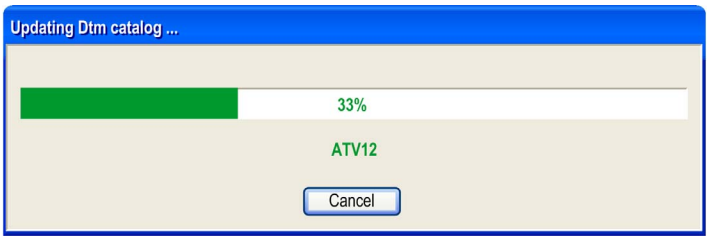
- automatically when the Unity Pro software is launched in closed project mode
- or manually, from the **Hardware Catalog** window, anytime the customer requests an update.

When the Unity Pro software is launched, in closed project mode, and DTMs are installed or uninstalled on/from the PC, the DTM catalog must be updated using the following procedure:

| Step | Action |
|---|---|
| 1 | When DTMs are installed or uninstalled and the Unity Pro software is launched.<br>**Results:** The dialog box update opens:<br>  |
| 2 | Click on the **Yes** button.<br>**Results:** When the update is completed, a message box appears: |

**NOTE:** Only a V1.2.1 FDT/DTM version or higher provides the detection service.

The DTM catalog can be updated manually update can be done using the following procedure:

| Step | Action |
|---|---|
| 1 | From the hardware Catalog *(see page 260)*, click on the **Update** button.<br>**Results:** The dialog box update opens: |
| 2 | Click on the **Yes** button.<br>**Results:** The standard Unity Pro progress control dialog is displayed:<br>  |

Updating the **Hardware Catalog** does not modify the build status of the DTM. It creates an internal catalog file based on the installed DTMs existing in the Windows® registry.

The time necessary for an update depends on the number for DTMs installed and their sizes.

## DTM Version Compatibility

The compatibility of DTM versions is managed by the vendor. They cannot be managed in Unity Pro.

During an update no check is made if a DTM missing from the **Hardware Catalog** is used in an application.

There are 2 methods to check and display any inconsistencies:

- Build the whole Unity Pro application
- Run the Check devices *(see page 273)* service from the **HostPS** contextual menu

# Section 7.3
# DTM Browser

## About this Section

This section describes the DTM Browser, which enables the user to select the different DTMs that make up the DTM connectivity tree on the host PC.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| DTM Browser | 263 |
| Ethernet Ready Equipment | 268 |
| Ready, Built, Connected and Installed DTMs | 270 |
| DTM Browser Contextual Menus | 273 |
| Field Bus Discovery Service | 281 |
| Store Device Configurations | 284 |
| Renaming DTMs | 285 |
| Profibus DTMs | 287 |

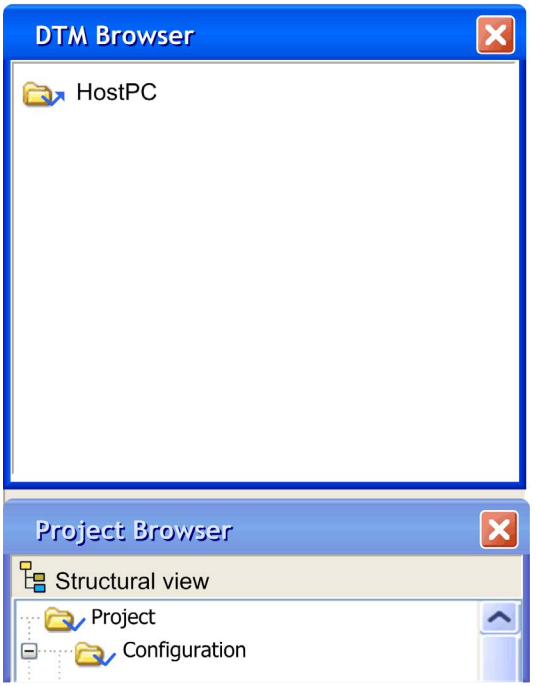
# DTM Browser

## At a Glance

The DTM Browser is the main component of the Unity Pro FDT Container. The DTM devices are managed from the browser.

## DTM Browser

To open the DTM Browser use the **Tools** menu -> DTM Browser:




The root of the DTM Browser is **HostPC**, which is the PC platform for the Unity Pro project.

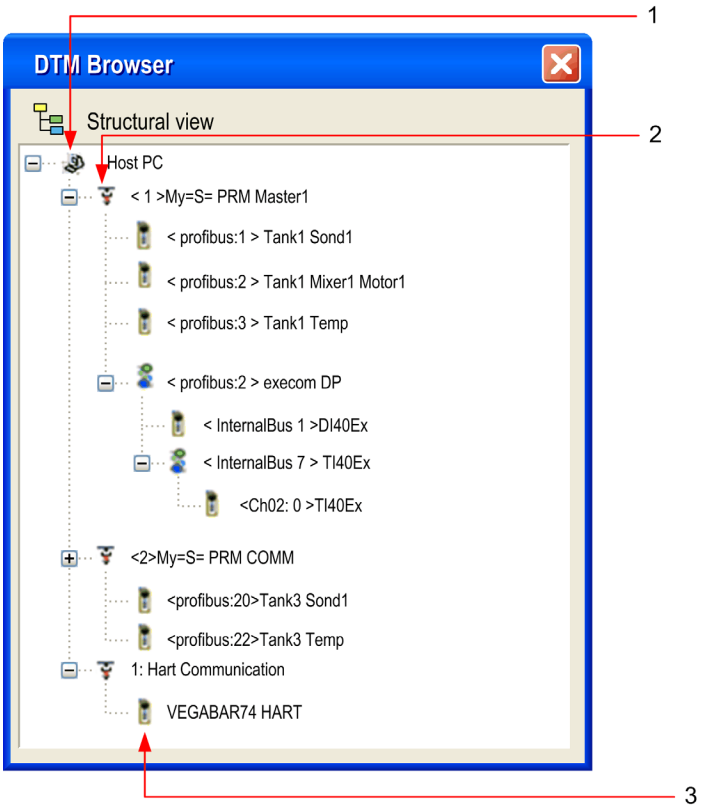
## DTM Browser DTM Types

The DTM Browser displays the configured DTMs added to the connectivity tree **HostPC** with hierarchical levels (nodes).

There are 3 types of DTMs:

- Communication DTMs (drivers)
  - Any COM DTM can be plugged directly under the root node (HostPC) and must be at the 1st level
  - A COM DTM can support Gateway DTMs or Device DTMs as children if their protocols are compatible
- Gateway DTMs
  A Gateway DTM can support other Gateway DTMs or Device DTMs as children if their protocols are compatible.
- Device DTMs
  A Device DTM does not support any child DTMs

The following illustration is a connectivity tree:



The following table describes the different hierarchical nodes of the connected DTMs in the above example connectivity tree:

| Number | Description of Level |
|---|---|
| 1 | The First Level is the root node, HostPC, which is where the PC is connected to the physical network. |
| 2 | The Second Level devices are communication DTMs (drivers) that connect to their physical communication devices on the network. |
| 3 | The Third Level devices are connected to a communication device. |

## Copy Paste feature

Each device DTM configured can be copied and pasted to easily create configuration.

**NOTE:** Only Device DTMs can be copy and pasted.

Operating mode to make a Copy/Paste:

- Make a right click on the Device DTM you want to Copy/Paste
- Select **Copy**
- Make a right click on the Communication DTM or Gateway DTM in which you want to paste the Device DTM
- Select **Paste**.

**NOTE:** If protocols are matching only device DTM can be paste under Communication DTM, only if protocols are matching. Communication DTM has to be in off-line mode.

**NOTE:** If protocols are matching, only device DTM can be paste under Gateway DTM, only if protocols are matching. Gateway DTM has to be in off-line mode.

**NOTE:** When a DTM is pasted in the connectivity Tree, the General tab *(see page 289)* is displayed. By default, the copied DTM alias is based on a valid DTM alias name.

**NOTE:** A default address is automatically created after the paste of a device DTM. The address can be modified when entering in the father DTM configuration screen. Copy/Paste feature has limitation, it can only be done inside an application and can not be done between two applications.

### Tree View Naming

Each DTM has a default name when inserted into the browser. The syntax is the following:
<Channel: Bus Address> User Name:

- Channel
  This is the name of the **channel communication media** where the DTM is plugged in. This name is read from the DTM and is set by the device vendor.
- Bus Address of the **device**. It can be either:
  - The connection point on its parent gateway network
  - The slot number in the modular device parent internal bus

  This information is supplied by the parent DTM and can be modified in Unity Pro if the DTM supports this modification.

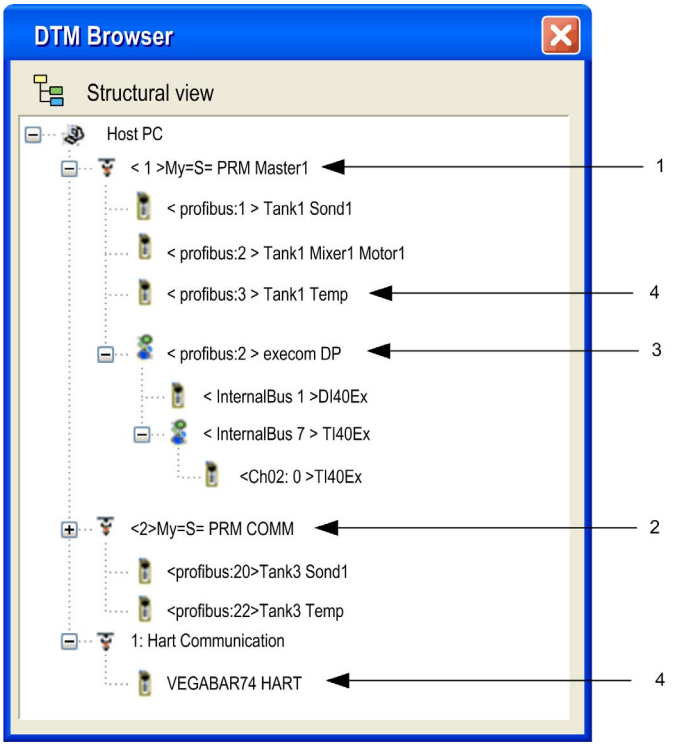
- User Defined Name
  This name is set by default to the vendors name for the DTM name (which is the default vendor name), but can be reset *(see page 285)*.

Examples:

- For a communication DTM
  <1> My =S= PRM COMM, where 1 is the references number.
- For a Device or a Gateway
  <Profibus:1> Tank1Sond1 means that the DTM is connected to the Profibus channel of its “parent” at connection point 1.

## Example Connectivity Tree

The following illustration is a connectivity tree:




| Number | Node Level Description |
|---|---|
| 1 | This Second Level PRM Comm Master communication DTM connects to the HostPC for communication with a PRM device via the Ethernet for DPV0 and DPV1 exchanges. |
| 2 | This Second Level PRM Comm communication DTM connects to the HostPC for communication with a PRM device via the Ethernet for DPV1 exchanges. |
| 3 | This Third Level Profibus Gateway DTM connects simple or Gateway devices to the higher communication DTM. |
| 4 | The Simple or Compact DTMs are connected to communication DTMs that allow them to communicate with their physical devices on the network. These tree leaves do not allow any sub-connectively. |

# Ethernet Ready Equipment

## Definition

An Ethernet Ready equipment is an Ethernet equipment that provides additional services compared to standard EIP or Modbus equipment.

## Applicability

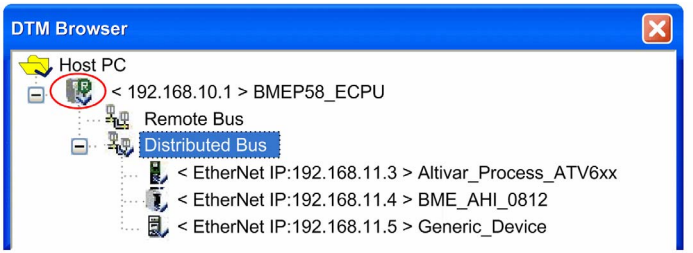
This concept of Ready equipment applies to:

- Standalone devices connected to the Ethernet I/O
- Modules plugged on the Ethernet backplane

**NOTE:** The only Ready module that exists is the BME CXM 0100.

## Representation

In Unity Pro, such devices are identified with a specific green label Ⓡ in the DTM browser:



## Services

Ready services are related to the software integration of the products into Unity Pro. It is based on DTM technology.

This table presents the services available for Ethernet Ready equipment:

| Service | Description |
|---|---|
| Single parameter entry | Any device parameter that you enter is shared into the entire configuration environment. |
| Bus editor declaration | Declaration is done from the Ethernet I/O bus editor or PLC bus editor in the Unity Pro browser. It uses products that belong natively to the Unity Pro Hardware Catalog. |
| System transfer | You can transfer all the system in a single operation: the Ready device configuration is transferred to the PLC with the PLC application.<br>**NOTE:** The PLC acts as a configuration server for all the Ready devices. |
| Scanning capacity | Scanned in a deterministic way (from the CPU RIO scanner(1)): you can attach the Ready equipment to any PLC task. It is refreshed within the task scan.<br>**NOTE:** This Ready equipment is called remote equipment to differentiate it from distributed equipment, which is scanned from the DIO scanner (RPI can be changed and is not linked to the task period). |
| Alert message | Inform you when a modification done from the device DTM impacts the PLC application and can lead to stop the PLC. |
| Online help integration | – |
| Shared user rights | User rights are automatically shared between Unity Pro and the device DTM. |
| (1) Scanners that accept Ready devices (M580 CPU) are also identified with this green label  . | |

# Ready, Built, Connected and Installed DTMs

## At a Glance

The status of a DTM is indicated in the DTM Browser: Ready, Not-Built, Connected and Installed.

## Ready

**Ready** devices or modules are devices or modules that are used by DTMs and fully integrated in the Unity Pro software, they mainly bring two particularities:

- They use new Unity Pro mechanisms to be integrated into the Unity Pro hardware catalog.
- They are managed by Unity Pro Ethernet I/O bus editors as for any X80 I/O modules.

A red “R” icon indicates that the device or module is a **Ready** one.

## Built or Not-Built DTMs

Build status:

- **Not-Built**: A blue check marks indicated that a node or one of its sub-nodes have changed, the information stored in the physical device is no longer consistent with the local project.
- **Built**: After building the Unity Pro project, the blue check mark is no longer in front of browser nodes that were changed before the build.

## Connected or Non-Connected DTMs

Connecting a DTM to its physical device automatically connects all higher level parent nodes up to the root node (all DTMs that the DTM being connected depends on.

Disconnecting a DTM from its physical device automatically disconnects all its lower level child nodes (all DTMs that depend on the DTM being disconnected).

Connection status:

- **Connected**: Connected nodes are represented by **bold** text in the browser. A node can be connected without all its sub-nodes being connected.
- **Non-Connected**: Non-connected nodes are represented by **normal** text.
  **NOTE:** Connecting a fieldbus or device is not linked the Unity Pro connection to a PLC (Off-line or On-line). DTMs can be connected/disconnected while Unity Pro is either Off-line or On-line.

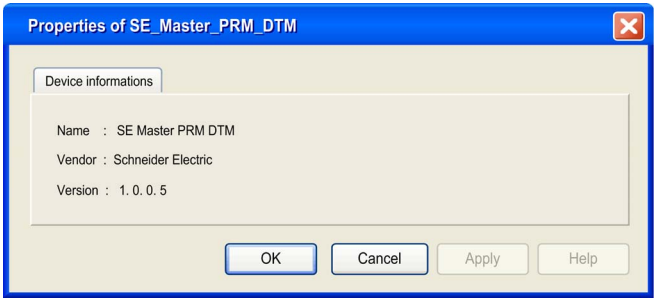
## Installed DTMs

Installation status:

A red X on the icon of a DTM indicates that the DTM software is not installed on the PC.

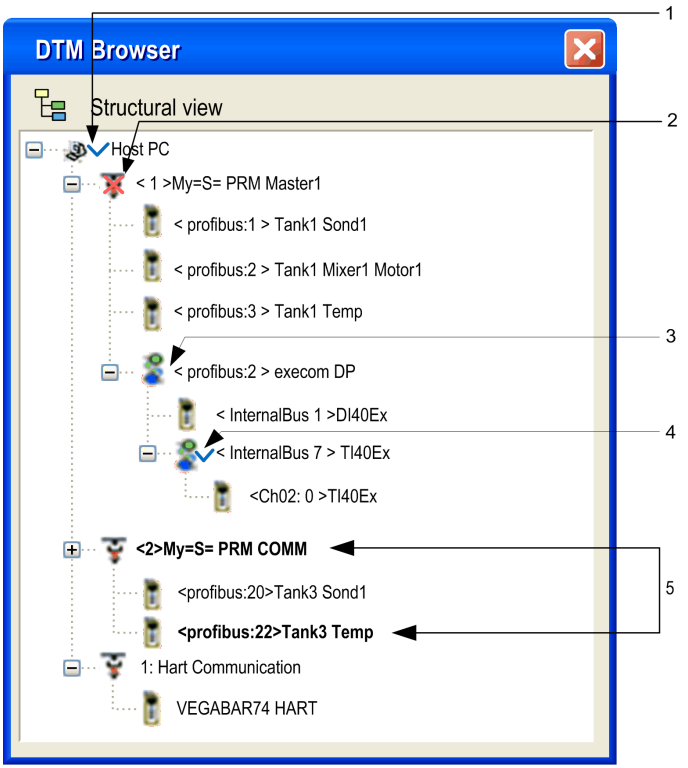
- This status is updated (red X **added**) when devices are detected but no DTM software is installed on the Host PC when the user:
  - Updates the **Hardware Catalog**
  - Tries to build the application
  - Tries to use a service in one of the 2 right-click contextual menus in the DTM Browser
  - Tries to use the Open service by left-clicking twice on a DTM in the DTM Browser
- This status is updated (red X **removed**) when the user updates the **Hardware Catalog** with the missing DTM.

A right-click on an invalid or non-installed DTM opens a 2-item contextual menu:

| Function | Description |
|---|---|
| Delete | This removes the selected DTM (and its sub-nodes) from the DTM project. |
| Properties | To help identify the non-installed DTMs, this function displays one tab (Device information) in the Properties of dialog box:  |

## DTM Browser

The following illustration gives examples of built/non-built, connected/non-connected and non-installed statuses:




The following table explains how a DTM Browser indicates if the DTMs are built, connected and non-installed:

| Number | Status | Number | Status |
|---|---|---|---|
| 1 | Not-Built | 4 | Not-Built |
| 2 | Non-installed | 5 | Connected |
| 3 | Built | | |

# DTM Browser Contextual Menus

## At a Glance

The (right-click) contextual menu of the DTM Browser offers functions that depend on item clicked on.

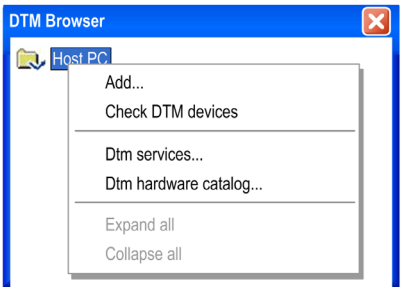
**NOTE:** The contextual menu items are also available via the **Edit** and **View** menus.

The DTM Browser Contextual Menu accesses:

- Adding and deleting DTMs
- Connecting and disconnecting DTMs to their physical devices
- Displaying and printing the properties of a DTM
- Transferring DTM configuration information to and from the physical device
- DTM-specific functions (via the Device menu function)
- Field Bus Discovery used to scan the physical devices in a fieldbus network

## HostPC

The **HostPC** contextual menu:




The **HostPC** contextual menu has the following items:

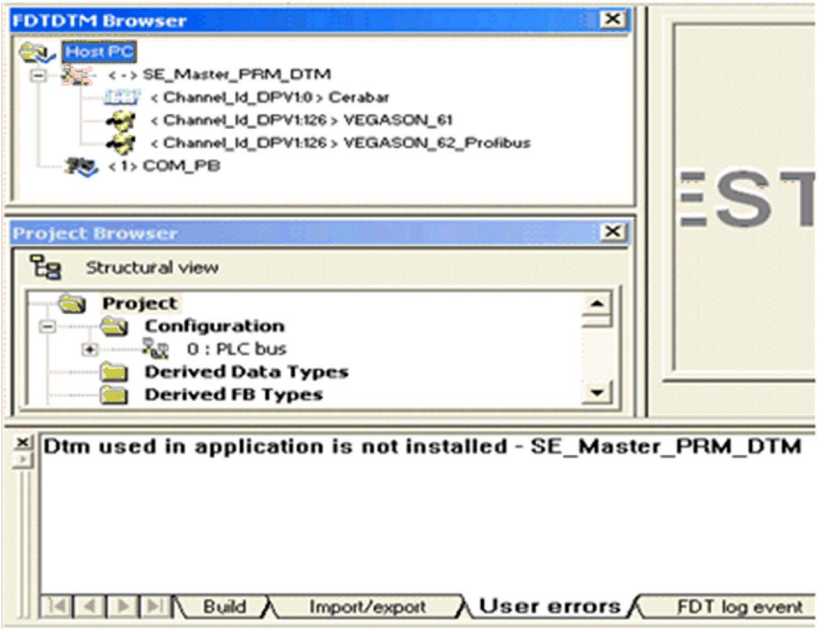
| Name | Description |
|---|---|
| Add device[1] | The **Add**, a version of the **Hardware Catalog**, opens allowing the selection of a Communication DTM. |
| Check DTM devices[1] | This function checks the current project for invalid or non-PC-installed DTMs. Refer to Check DTM devices *(see page 274)* below. |
| DTM Services[1] | This function allows the users to select/unselect all the DTMs added in the project and performed actions on the selected DTMs. Refer to DTM Services *(see page 275)* below. |

| Name | Description |
|---|---|
| DTM Hardware Catalog[1] | This function open the DTM catalog tab in the Hardware Catalog *(see page 257)*. |
| Expand all[2] | All DTMs in the project are shown. |
| Collapse all[2] | Only the Commutation DTMs in the project are shown. |

Legend:
**1** This function is available via the **Edit** menu.
**2** This function is similar to Project Browser Expand/Collapse all *(see page 170)*.

## Check DTM Devices

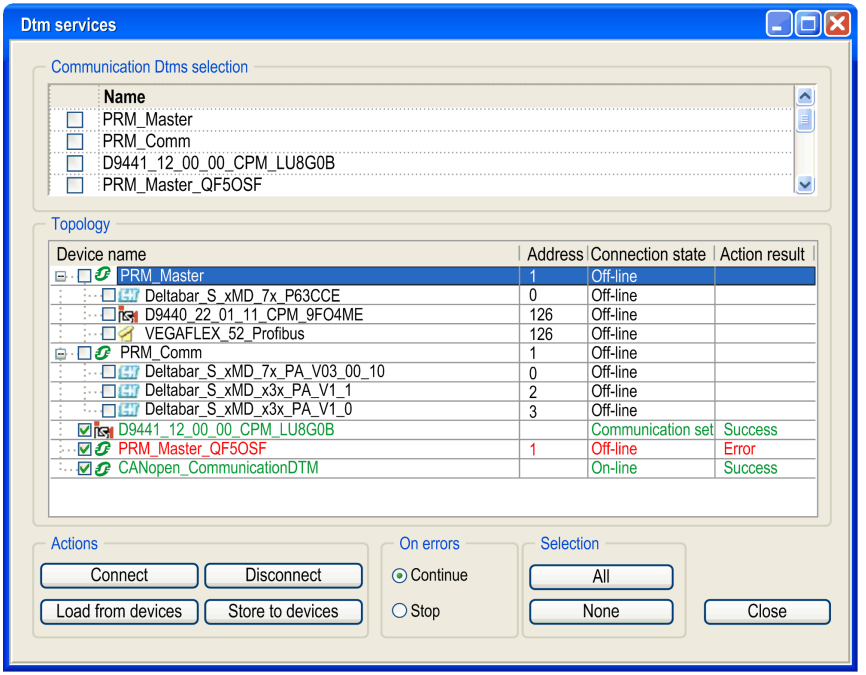
If the results of the check include invalid or non-installed DTMs, they are displayed in the **User errors** tab in the bottom information window and a **Red X** is placed over their icons in the DTM Browser:




**NOTE: Check DTM devices** only detects installed DTMs, for GSDs verify the GSD library.

## DTM Services

The users can select/unselect DTMs added in the project and performed actions on the selected DTMs:




The **DTMs services** dialog box menu contains the following group boxes:

| Name | Description |
|---|---|
| Communication Dtms selection | The user can select or unselect all communication DTMs and attached DTMs currently added in the project. |
| Topology | The **Topology** area lists all DTMs added in the project. The set of selected DTMs can be modified by selecting or unselecting the DTMs in the Structure tree control. |
| Actions | The user can perform actions for each DTMs contained in the set of DTMs previously selected in the Structure tree control. |
| On errors | The user can select to **Continue** or **Stop** the process if an error is detected. A dialog box is displayed whatever options is selected. |
| Selection | The user to select either all DTMs added in the project or none. |

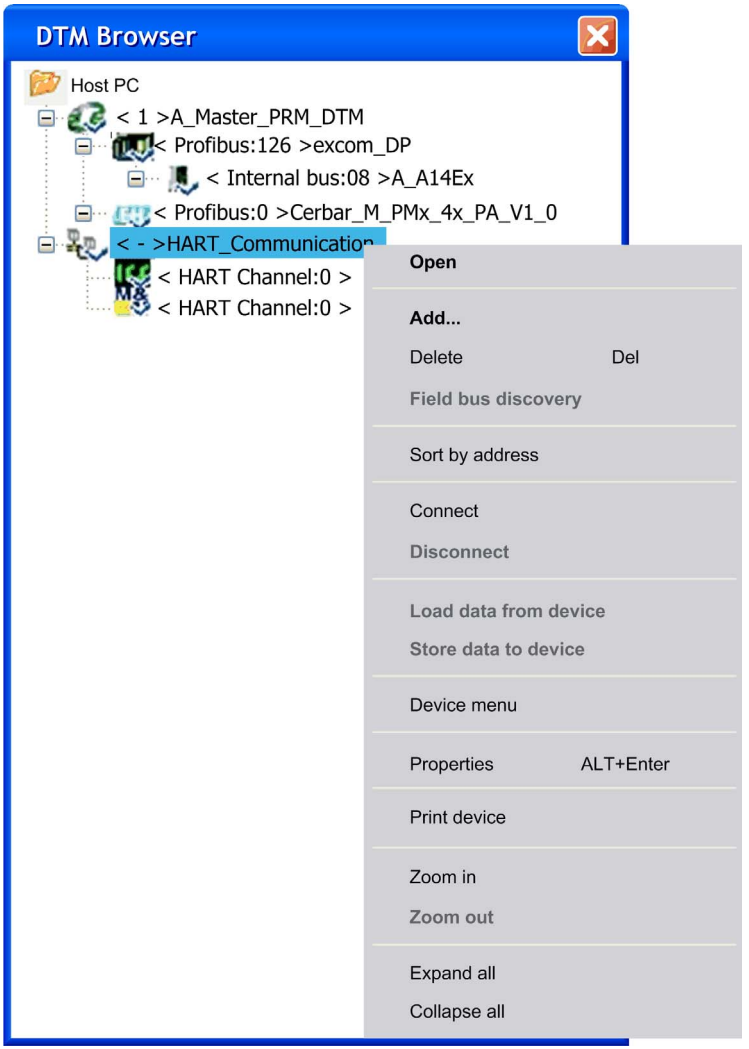
The dialog box below is an example when the user selects either **Continue** or **Stop** in the **On errors** group box:




**NOTE:** The entire process can be cancelled any time by using control button Cancel.

## Communication and Other DTMs

The available DTM Browser contextual menu items differ according to the DTM selected:



The **DTM** contextual menu has the following items:

| Name | Description |
|---|---|
| Open[1] | This opens the “<device name> - DTM Configuration” window with the Configuration and Device Description information supplied by the device vendor.<br>A double-click on the DTM in the DTM Browser also opens this window. |
| Add...[1] | This opens the **Add** dialog, a version of the **Hardware Catalog**, allowing the instantiation of a DTM.<br>The DTMs are filtered, that is, the **Add** window only displays DTMs compatible with the DTM selected in the DTM Browser.<br>To add a device refer to Add device *(see page 280)* below |
| Delete[1] | If the selected DTM allows this function, this deletes the selected DTM and its sub-node DTMs from the DTM connectivity tree.<br>Deletion from the DTM connectivity tree does not effect the DTMs link to the I/O Scanning table. |
| Field Bus Discovery | This scans the connected physical devices to create the corresponding field bus connection topology. This item is only available in the contextual menu if a device connected.<br>Refer to Field Bus Discovery *(see page 281)* service.<br>**NOTE:** The DTM to scan has to be installed in the PC and the **Hardware Catalog** must be up to date. |
| Sort by address | This displays the DTMs sorted under their parent DTMs according to the bus address of its physical device in ascending order. |
| Connect[1] | This connects the DTM to its physical device on the network.<br>This connection does not depend on the PLC online/offline status of the Unity Pro project application.<br>Connecting a Gateway- or Device-DTM implicitly connects its parent DTM.<br>While the device is connected, **Connect** is not available. |
| Disconnect[1] | This disconnects the DTM from its physical device. This disconnection does not depend on the PLC online/offline status of the Unity Pro project application.<br>Disconnecting a DTM implicitly disconnects all its child DTMs.<br>While the device is disconnected, **Disconnect** is not available. |
| Load data from device[1] | This uploads the configuration parameters from the physical device to the local DTM, but not from the application (binary) in the PLC.<br>While the device is disconnected, **Load data from device** is not available. |

| Name | Description |
|---|---|
| Store data to device[1] | This downloads the local DTM parameters to the physical device. Each DTM must be stored manually (one by one).<br>While the device is disconnected, **Store data to device** is not available.<br>To make an application in the PLC consistent with the stored DTM, the user must build the Unity Pro application and download the new version of the application to the PLC |
| Device Menu | The functions in this menu are vendor dependant, such as:<br>● Compare (the online and offline parameters)<br>● Diagnostic (displays a diagnostic screen)<br>● Print (HTML)<br>● Observe<br>● Additionnal functions available when selecting the communication DTM:<br>❍ EDS (add or remove from librairy)<br>❍ Online Action<br>❍ Explicit Message (Ethernet or Modbus)<br>❍ About<br>❍ Advanced Mode[4]<br>When configuring a DTM, the changes are validated in Unity Pro in 2 ways, depending on the changes:<br>● Automatically as the changes are made.<br>● You must click on a **OK**, **Apply** or **Validate** button after making the changes. |
| Properties | Refer to the DTM Properties Tabs *(see page 288)* section. |
| Print device | If this optional function is supported by a DTM, it opens the vendor device documentation in Internet Explorer, which can then be printed.<br>Only one DTM information can be printed out at time. A print of the whole network is not available. |
| Zoom in[1],[2] | This displays only the selected DTM node and its child DTMs.<br>While zoomed-in, **Zoom in** is not available. |
| Zoom out[1],[2] | This returns to the full display of the DTM connectivity tree.<br>While there is a full display, **Zoom out** is not available. |
| Expand all[3] | This displays all DTMs below the selected DTM. |
| Collapse all[3] | This displays only the selected DTM. |

Legend:
**1** This function is available via the **Edit** menu.
**2** This function is similar to Project Browser Zoom in/out *(see page 167)*.
**3** This function is similar to Project Browser Expand/Collapse all *(see page 170)*.
**4** Advanced option are available DTM configuration screen

## Add a DTM Device

To add a new DTM to the connectivity tree in the DTM Browser:

| Step | Action |
|---|---|
| 1 | Make sure that the DTM appears in the **Hardware Catalog**.<br>If it is not in the catalog, perform an update *(see page 260)* of the **Hardware Catalog** using the **Update** button.<br>If it still does not appear in the **Hardware Catalog**, the DTM must be installed on the PC *(see page 258)*. |
| 2 | In the DTM Browser select the node where the DTM is to be added. |
| 3 | Right-click and select **Add...** |
| 4 | In the **Add** window select the DTM to be added to the DTM connectivity tree. |
| 5 | Either:<br>● Click on the **Add DTM** button<br>● Double-click on the selected DTM<br>Results: The 4-tab DTM **Properties of** dialog opens. |
| 6 | Using the 4 tabs, verify that the correct DTM has been added to the DTM connectivity tree and verify its configuration. |
| 7 | If the DTM Alias name does not meet the Unity Pro (or the user application) naming rules, it must be changed before closing the dialog. |
| 8 | Click on the **OK** button to finish adding the DTM to the network tree or click on the **Cancel** button to close the dialog without adding anything. |

# Field Bus Discovery Service

## At a Glance

The Field Bus Discovery service is available for Communication and Gateway DTMs that:

- Support device scanning
- Are connected to their physical devices

Only the first level devices below the Communication and Gateway DTMs are detected (there is no recursive scanning).

## Field Bus Discovery Use

The results of the scanning process is compared to the registered DTMs in the DTM catalog of the computer. If a match is found in the DTM catalog for a scanned device, the results are accompanied with a matching type that gives the accuracy of the match.

The three available matching types are:

- Exact Match:
  All identification attributes are matching. The correct device type was found.
- Generic Match:
  At least the Manufacturer ID and Device Type ID attributes match. The support level of DTM is “Generic Support”.
- Uncertain Match:
  At least the Manufacturer ID and Device Type ID attributes match. The support level of DTM is **not** “Generic Support”.

The following procedure explains how to use the Field Bus Discovery service:

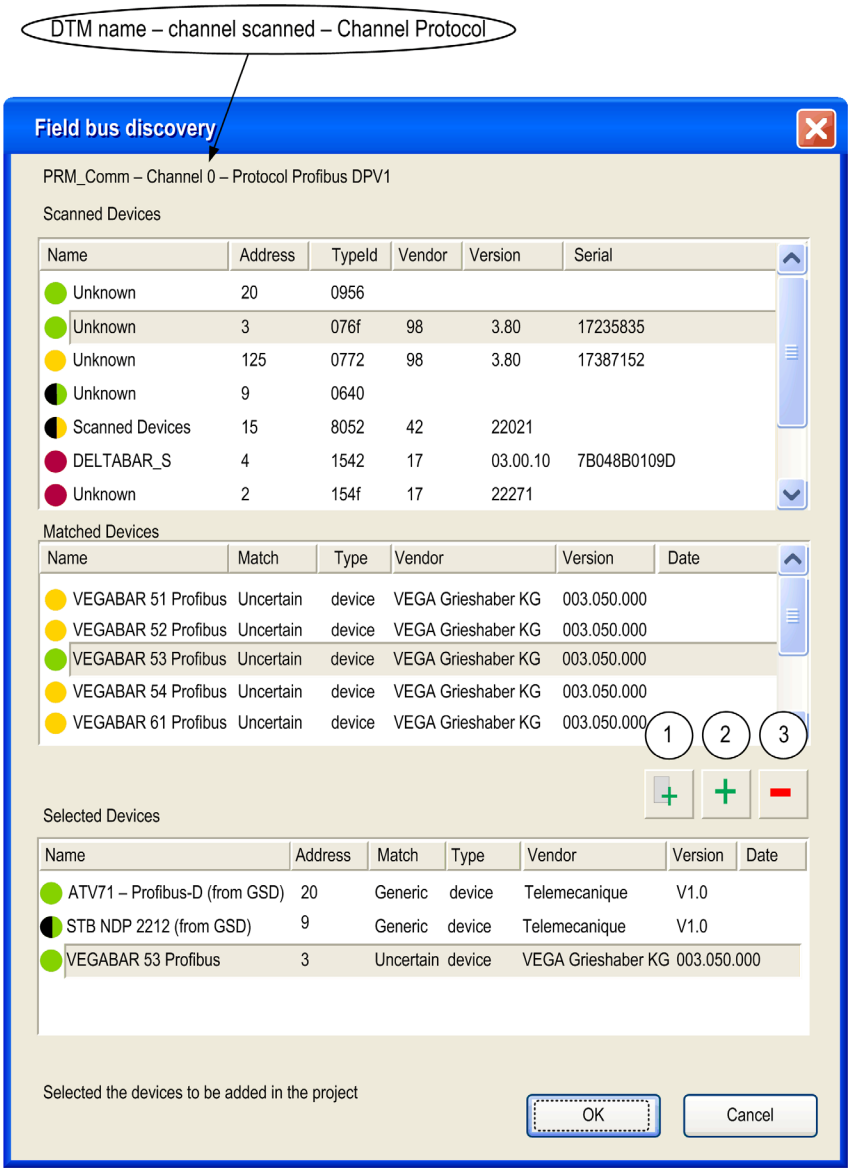
| Step | Action |
|---|---|
| 1 | In the DTM Browser select an appropriate DTM. |
| 2 | Right click to open the Contextual menu **Field Bus Discovery**. |
| 3 | If necessary, select the channel and protocol:<br>● If the DTM has more than one channel<br>● If the channel supports more than one protocol |
| 4 | Click on **OK** in the message box to have this service detect all devices on the selected channel. |
| 5 | If at least one matched device has been found, the Field Bus Discovery dialog *(see page 282)* is displayed listing the scanned and matched devices. Select the matched devices to be created in the Unity Pro project. |

**NOTE:** a device that is connected on the bus can be detected if:

- Its DTM is installed on the PC.
- Its GSD is added into the GSD library (if not a DTM device).
- The DTM hardware catalog is up-to-date (with the DTM installed and the GSD in the library).

## Field Bus Discovery Dialog

If at least one matched device has been found, the Field Bus Discovery dialog box is displayed listing the scanned and matched devices. Select the matched devices to be created in the Unity Pro project (which then shows up in the **Selected Devices** list:



This dialog has three lists:

| List | Description |
|---|---|
| Scanned Devices | This list displays all the devices (matched and unmatched) found during the scan. |
| Matched Devices | This list displays the matched DTM found in the workstation DTM catalogue for the selected device in the **Scanned Devices** list.<br>Each time a scanned device is selected in the **Scanned Devices** list, the contents of the **Matched Devices** list is updated to display the matched device DTM found for the selected scanned device.<br>The matching process can yield one or more matched devices for a given scanned device. In this case, select one among the matched DTMs found. |
| Selected Devices | This list displays the device DTMs selected to be added to the Unity Pro project. |

The lists use colored icons:

| Color | Description |
|---|---|
| Green | Indicates that the device has been selected |
| Yellow | Indicates that the device has been matched |
| Red | Indicates that the device has **not** been matched |
| Black | Indicates information about the address of the scanned device:<br>● In the **Scanned Devices** list, the device has an address identical to one of the DTMs in the Unity Pro project<br>● In the **Matched Devices** list, the device will be assigned an address identical to one of the DTMs in the Unity Pro project |

This dialog has five buttons:

| Button | Use this button to... |
|---|---|
| 1 | automatically add the best matched (according to the matching types listed above) device DTM for each found device in the **Matched Devices** list to the **Selected Devices** list. |
| 2 | add the matched device DTM selected in the **Matched Devices** list. |
| 3 | remove one or more devices from the **Selected Devices** list. |
| OK | insert the device DTMs in the **Selected Devices** list into the Unity Pro project.<br>If there are one or more devices in the **Selected Devices** list that have the same address in the Unity Pro project, a message box opens asking if you want to continue.<br>If you click **OK**, all devices in the Unity Pro project that have identical addresses as the selected devices is **deleted** and **replaced** by the DTMs selected in the **Selected Devices** list. |
| Cancel | cancel the Field Bus Discovery scan and do nothing. All information in the three lists is lost |

# Store Device Configurations

## Introduction

After configuring or modifying a configuration of a DTM, the changes must be downloaded to its physical device using Store data to device *(see page 273)*.

## Connectivity Tree

For a connectivity tree it is necessary to:

- Store the configuration for each DTM into the corresponding device so that the device works correctly
- **Build** or **Rebuild all** the Unity Pro application and download it to the PLC be able to upload the complete application including all DTM configurations from the PLC binary

The total number of operations equals the number of DTMs stored + 1 download of rebuilt application.

## Recovering Connectivity from the PLC Application

If the complete application has been stored in the PLC (ETS option checked), it includes the connectivity of the DTMs.

Uploading the application from the PLC provides the connectivity stored in the PLC. For this, the PC used for uploading must have the corresponding DTMs *(see page 266)* installed.

Verify that the corresponding configuration is stored in the devices to avoid de-synchronization of the project..

## De-synchronization of Project

If a DTM configuration is changed and stored to its device, the application running in the PLC is not updated.

To synchronize the PLC application with the device configuration, rebuild the Unity Pro application to take into account the changes and download it to the PLC.

Uploading a binary from a PLC overwrites the configurations of all DTMs in Unity Pro applications retrieved from the PLC. Any modifications of the DTM configurations that are not stored in the PLC binary are overwritten by the configuration in the PLC binary and lost.

| ⚠ CAUTION |
| --- |
| **UNEXPECTED EQUIPMENT OPERATION**<br>Do not upload a project from the PLC if any changes have been made in any devices connected to the PLC since the last build and download of the project.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

# Renaming DTMs

## At a Glance

According to the FDT standard, the official name of a DTM is its Tag name assigned by the DTM vendor.

Unfortunately, some DTMs have Tag names that do not:

- Allow the Tag name to be changed
- Meet the Unity Pro naming rules

The solution is to choose the DTM Alias name that meets these rules (and any special user defined application naming rules).

## Alias Name

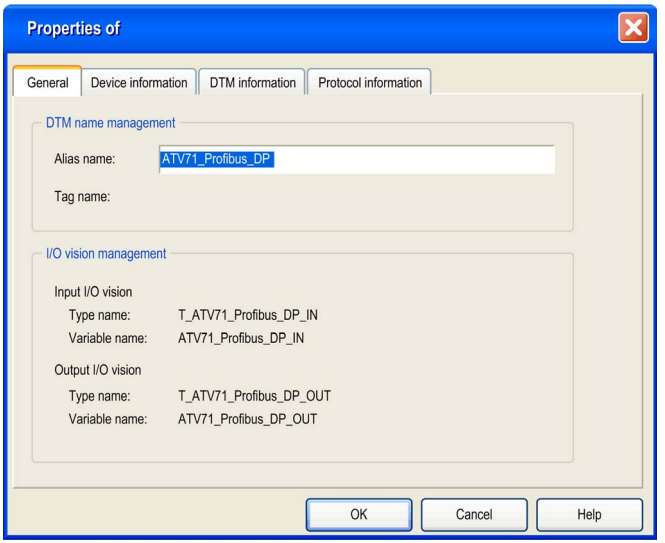
The Alias name is the DTM reference name in Unity Pro and must be unique in an application.

If possible, the default, the Alias name is the same as the DTM Tag name. But if necessary, the Alias name can be changed in 3 places:

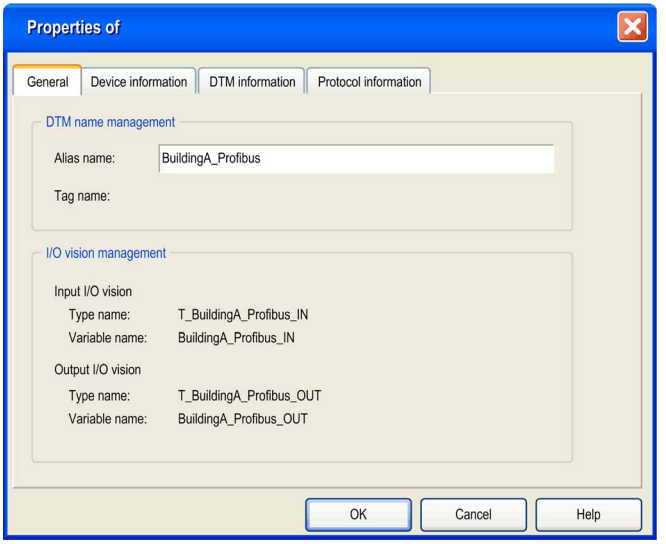
- Directly in the General tab *(see page 289)*
- In the DTM Browser by clicking on the selected DTM or using the F2 shortcut with the selected DTM.

## I/O Vision in Communication DTMs

For device DTMs attached to certain Schneider Electric Communication DTMs (such as the SE_Master_PRM_DTM), the default names of the DTM Input/Output Types (a structure) and the Variable instances of the Type are based on the default DTM Alias/Tag name:



Changing the Alias name automatically changes the names of the Input/Output Types and Variables:



# Profibus DTMs

## Configuration of Profibus Field Devices

Using vendor-supplied Profibus DTMs, users are able to remotely:

- Configure Profibus devices associated with the DTM
- Adjust the parameters of these devices while they are **off-line**
- Adjust the parameters of these devices while they are **on-line**

## PRM Master Bus Module and its DTMs

Unity Pro uses a Profibus Remote Master (PRM) Gateway module (the bus master physical device) and its PRM configuration tools (software):

- The PRM **Master** DTM that allows the PRM Gateway to carry out:
  - Explicit DPV0 functions (control and diagnostics of the Profibus slaves on the network).
  - Communication functions (implicit DPV1 exchanges with the Profibus slaves). It also provides access to the Profibus slaves from the Control network to HMI, SCADA or ASSET management tools.
- The PRM **Comm** DTM that only allows the PRM Gateway module to carry out communication of implicit DPV1 exchanges and access to management tools.

The Modbus TCP I/O scanning link between the PLC and the (PRM) Gateway module connected to the Profibus network is provided by a Quantum, Premium, M340, or M580 module with an Ethernet port.

# Section 7.4
# DTM Properties Tabs

## DTM Properties Dialog

### At a Glance

The number of tabs in a **Properties** dialog varies according where the dialog is called from.

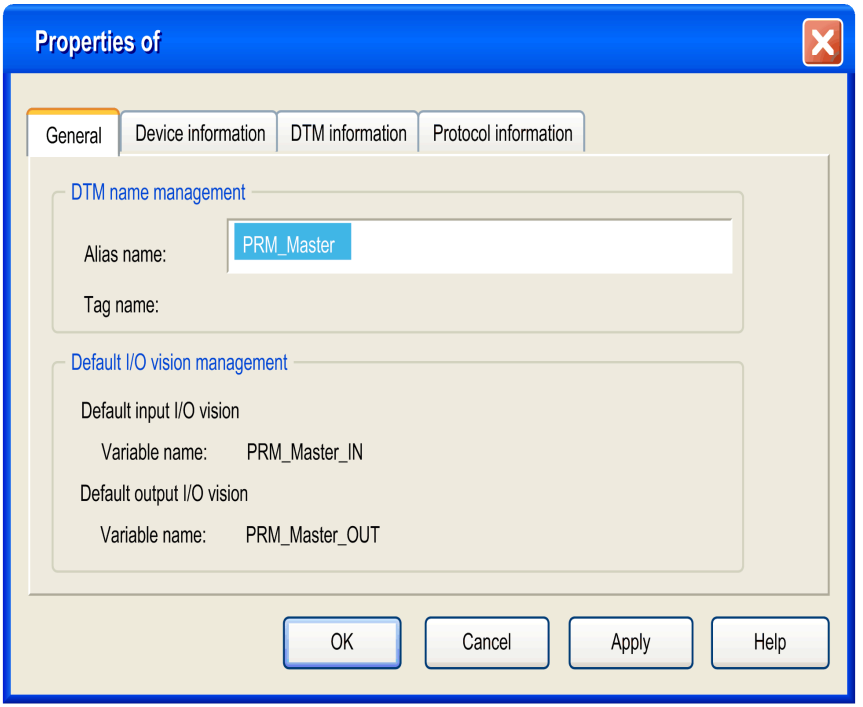
### Properties Tabs

The **Properties** dialog is called when a DTM is selected:

- To be added to connectivity tree in the DTM Browser from the **Hardware Catalog**:
- For the Contextual Menu **Properties** service in the **Hardware Catalog**:
- It opens with 4 tabs:
  - General
  - Device Information
  - DTM Information
  - Protocol Information

## General

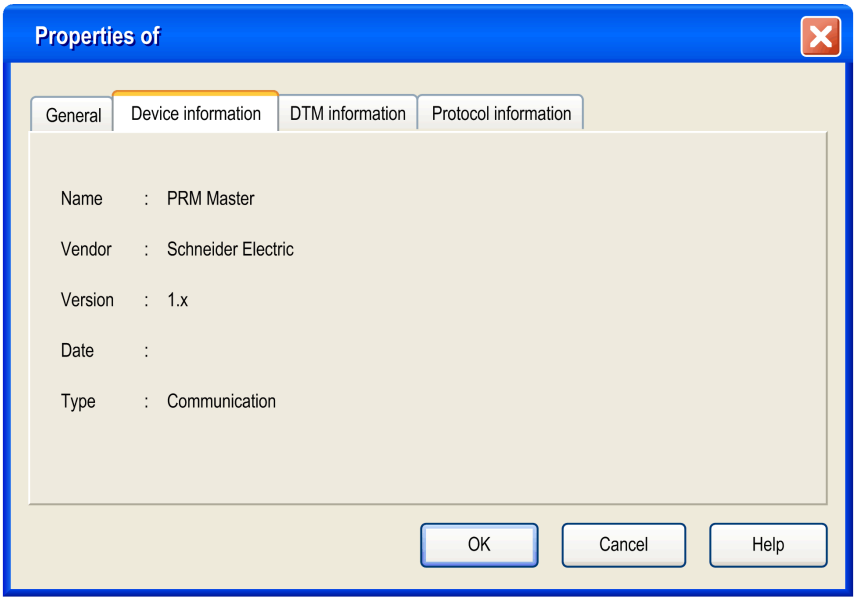
The General tab:




**NOTE:** The Alias name (and the DTM Tag name if the vendor allows it) can be changed *(see page 285)* here or in the DTM Browser.

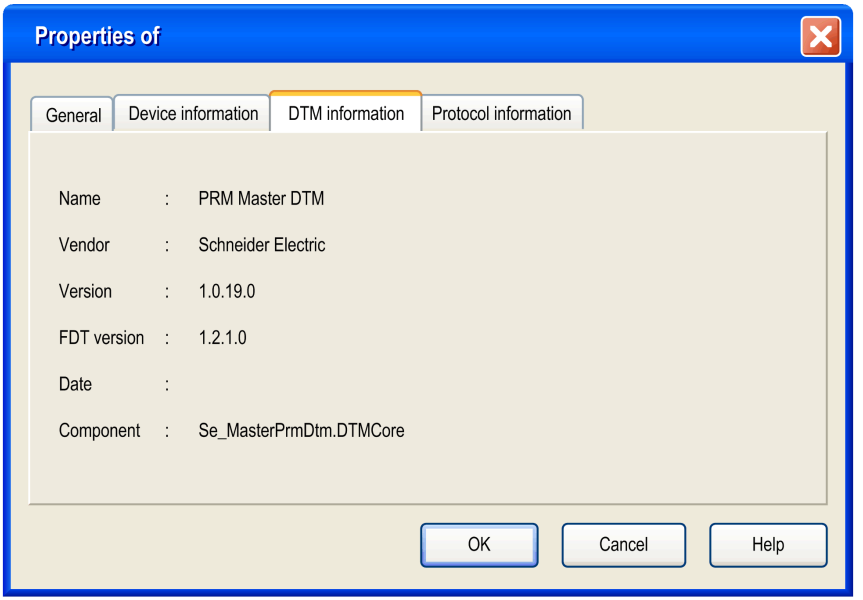
## Device Information

The Device Information tab:



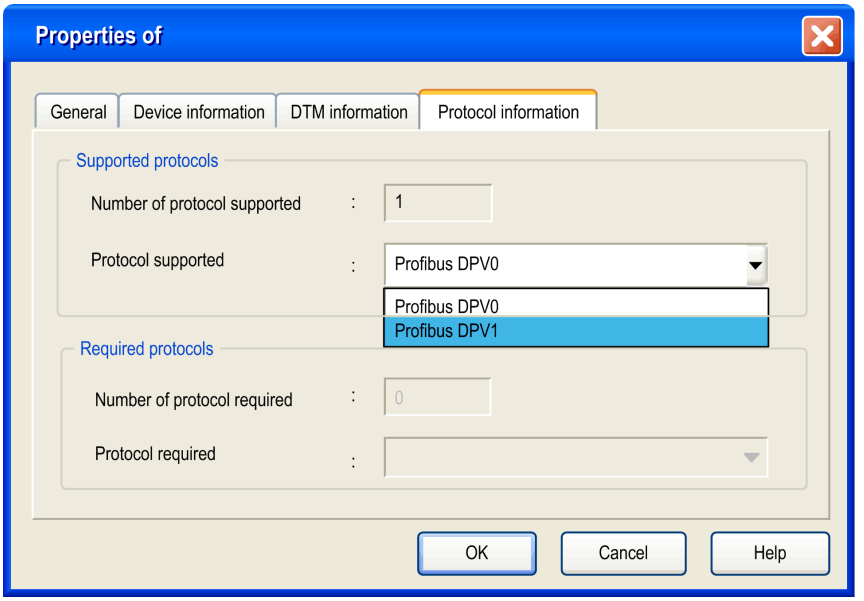
## DTM Information

The DTM Information tab:



## Protocol information

The Protocol Information tab:




| Item | Description |
|---|---|
| Supported protocols | These are the protocols available for the child nodes of this DTM. |
| Required protocols | These are the protocols that must be supported by the parent DTM of this DTM. |

# Chapter 8
# Type Library Manager

## Object of this section

This chapter describes the main functions of the Type Library Manager.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Libset (Set of Libraries) | 294 |
| Overview of the Type Library Manager | 297 |
| Type Library Manager | 298 |
| Description of the Type Library Manager | 301 |
| Loading an object from the library into the project | 307 |
| Transferring an object from the project into the library | 310 |
| Creating a new library and family | 312 |
| Creating an installable family | 314 |
| Updating a family of a library | 315 |
| Consistency Check Assistant | 317 |
| Library Version Management | 320 |
| Using the type library of an older Unity Pro Version | 324 |

# Libset (Set of Libraries)

## At a Glance

**Libset** (set of libraries) comprises all libraries, families, functions and variables (I/O data structures) that can be used to develop an automation project.

You can only modify the **Libset** version in the **Open** project dialog.

## Creating a New Project

Creating a new project, the latest **Libset** version is automatically selected for this project.

To modify the **Libset** version of your project, you have to save your project and open it again with the version of **Libset** you want to use.

## Opening a Project

In general, opening an existing project (STU file) or an archived project (STA file), you can select the **Libset** version.

Importing a ZEF or XEF file, the latest **Libset** version is always selected.

The following table describes the default **Libset** version used to open a project according to the file type.

| File Type | Project before Unity Pro 4.0 | Project from Unity Pro 4.0 to Unity Pro 5.1 | Project from Unity Pro 6.0 |
|---|---|---|---|
| STU file | Not applicable | The **Libset** version used to build the project is automatically selected. If the respective **Libset** version is not installed, the latest version will be used. | |
| STA file | The **Libset** version used to generate the STA file is automatically selected. If the respective **Libset** version is not installed, the latest version will be used. | The **Libset** version used to generate the STA file is automatically selected. If the respective **Libset** version is not installed, the latest version will be used. | |
| XEF file | The latest **Libset** version is selected. | The latest **Libset** version is selected. | |
| ZEF file | Not applicable | Not applicable | The latest **Libset** version is selected. |

## Modifying the Libset Version

Opening an existing project *(see page 87)* (STU file) or an archived project (STA file) you can select the **Libset** version.

The **Open** dialog provides the following options.

| Element | Description |
|---|---|
| **Keep project version** | Select this option to keep the **Libset** version used to build the project.<br>If the respective **Libset** is not installed, the latest **Libset** version will be used. |
| **Update with latest version available** | Select this option to update the project with the latest **Libset** version. |
| **Select a specific target version** | Select this option to choose a specific **Libset** version. You can select the version by means of the **Version** box below. |

If the **Libset** version is modified, the project is in modified state.

## Upgrade/Downgrade

- Upgrading the **Libset** does not update the project with the newly chosen libset automatically. EF/EFB differences will be displayed during opening the project in the **LIBRARY Version Management** dialog, where you can update the project.
  You can update the project later as well, using the contextual menu command in the **Type Library Manager** dialog *(see page 320)*.
  If you have updated the project, you have to start a **Rebuild All**. A **Rebuild All** needs to stop the PLC and to perform a global download.
- You can also downgrade the **Libset** version.
  There is no consistency check when downgrading the **Libset** while opening a project.
  So you may not be able to rebuild the project (using a block created in an earlier **Libset** version, not available in the latest **Libset** version).
  If the rebuild is unsuccessful, you have to save your project and to open it again with an appropriate **Libset**.
- If you want to add an FFB which is not yet contained in the current version of the **Libset**, you have to upgrade the whole **Libset**.
- A change of **Libset** version in the **Open** dialog is memorized in the project.

### Libset Information

If no project is opened, information is displayed on the latest **Libset**.

If a project is opened, information is displayed on the **Libset** used for the project.

Information on the **Libset** version used for your current project you can get from the **General** tab of the **Properties of Station** dialog.

In the **Project Browser** of Unity Pro right-click on **Station** and select **Properties** from the context menu. In the **General** tab you will find the **FFB Library version**.

Further on the following dialogs provide information on the **Libset** version (e.g. LibsetV3.0):

- **Types Library Manager**
- **Types Library Browser**
- **Data Selection**
- **Data Editor: Variable Type Selection**
- **Function Input Assistant: FFB Type Selection**
- **Data Properties**
- **PLCScreen → Information tab → APPLICATION → IDENTIFICATION**

### Types Library Manager

In the **Types Library Manager** click the **Information** button to get the following information:

- **Current library**
  - **Library status**
  - **Diagnostic**
- **Library contents**
  - **Families**
  - **Libraries**
  - **FFBs**

For the **Libset Information** window please refer to information *(see page 299)*.

# Overview of the Type Library Manager

## Principle

The Unity Pro Library contains all available objects to develop an automation project. These objects may be the following functions or variables:

- EF (Functions),
- EFB (Function blocks),
- DFB (user function blocks),
- DDT (Variables)
- ...

The Type Library Manager provides functions to edit the contents of the library and execute transfers between the library and the project you are working on.

Simple operations are automatically executed or you will be asked to confirm. An Assistant *(see page 317)* will support complex operations. This Assistant becomes available during loading, transfers, or deletions.

## List of available functions

The Unity Pro context menus provide access to the library management functions below:

- Creating a family or library
- Deleting a family or library
- Cutting a family or object
- Inserting a family or object
- Showing and hiding a tree structure for an object, family, or library
- Renaming the current entry in the tree structure
- Displaying the Assistant to load an object
- Fit columns
- Deleting types not used in the project
- Version check: Consistency between different versions of the objects of the library and project *Library Version Management, page 320*.

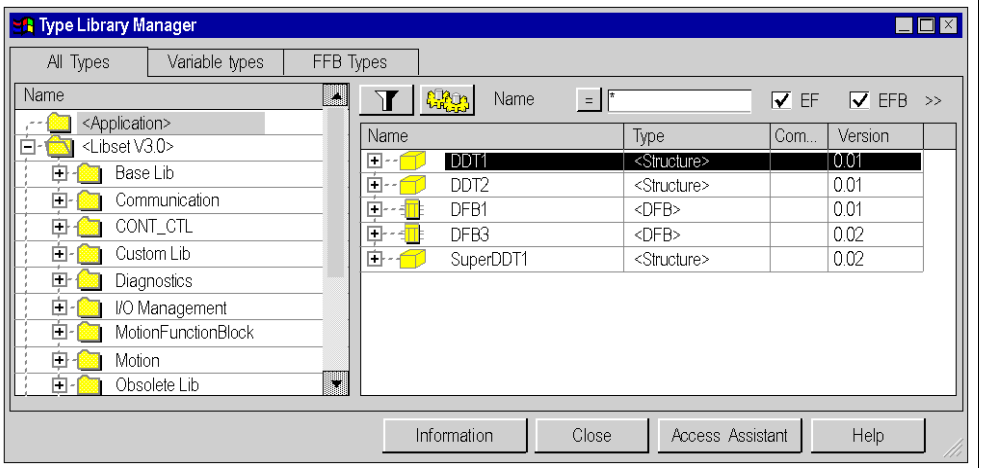
# Type Library Manager

## At a Glance

Unity Pro software provides a manager to edit objects in the library providing access to all managing functions of the library.

- Loading an object into the project from the library *(see page 307)*.
- Transferring an object of the project into the library *(see page 310)*.
- Deleting a library object
- Updating and managing library versions
- ...

## Accessing the Manager

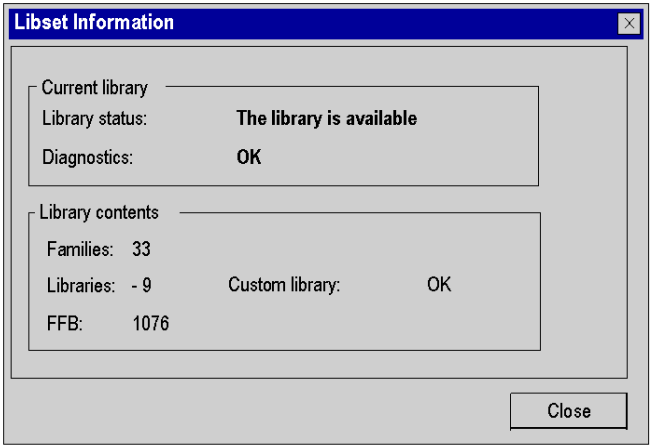
| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Manager** option.<br>**Result**: The following window appears.<br>  |

## Areas of the Type Library Manager

| Area | Description |
|---|---|
| Tabs | The tabs enable you to display the following according to your selection:<br>● All Types<br>● Variable types<br>● FFB Types (Functions) |
| The left sub-window | His sub-window shows the tree structures of the objects of the project, library of functions (division of functions by family), and the IODDT. |
| The right sub-window | This sub-window displays the list of objects included in the selection made in the left sub-window.<br>The check boxes above it are used to define this list. |
| Buttons | ● Information<br>Click this button to get information on the current library.<br>See paragraph below.<br>● Close<br>Click this button to close the dialog box without accepting changes.<br>● Access Assistant<br>Click this button to open a window to copy objects into a project (*Loading an object from the library into the project, page 307*).<br>● Help<br>Click this button to call up the dialog help. |

## Information

Click the **Information** button to open the **Libset Information** window.



The table below describes the various functions offered.

| Fields | Description |
|---|---|
| **Current library** | **Library status**: information on the status of the custom library<br>● available<br>● not available<br><br>**Diagnostics**: information on the validity of the library<br>● OK (valid)<br>● not valid. |
| **Library contents** | **Families:** indicates the number of families available<br>**Libraries:** indicates the number of libraries available<br>**FFBs:** indicates the number of function blocks available<br>**Custom Library:** indicates whether or not user libraries exist |

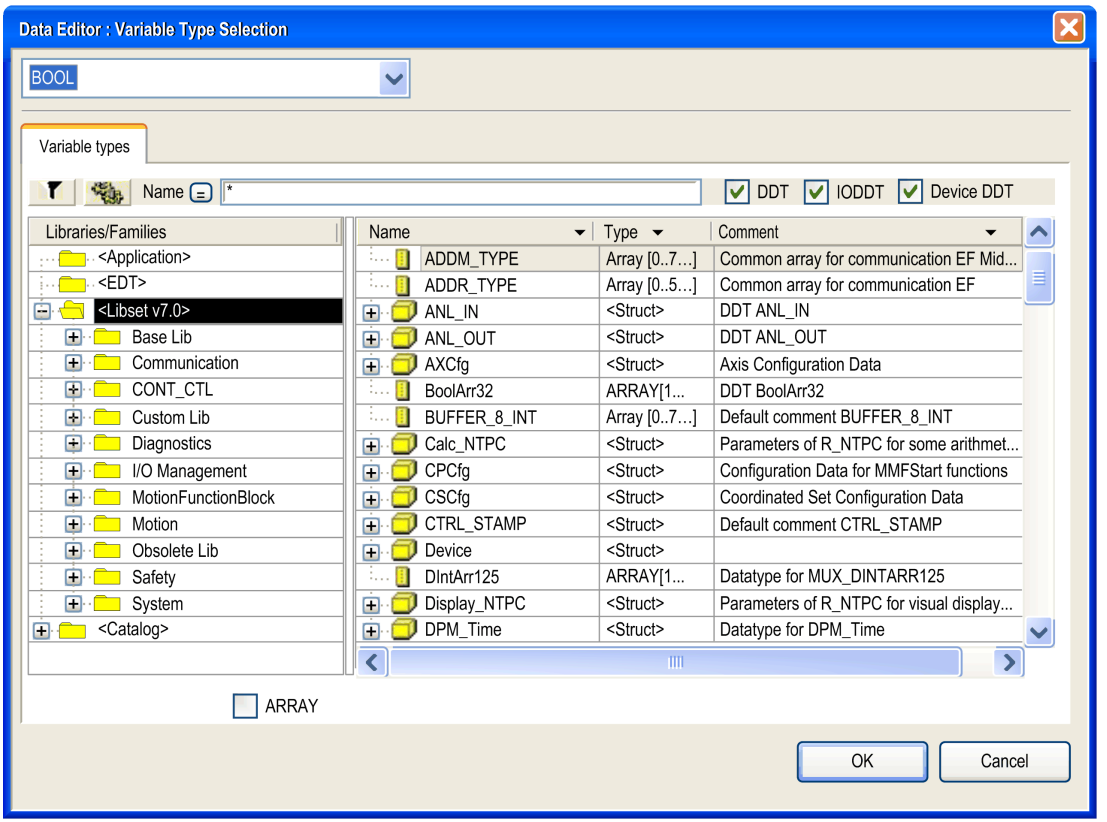
# Description of the Type Library Manager

## Dialog structure

The dialog box to select objects has three tabs:

- **All Types**
  Under this tab, you can select variables and FFBs.
- **Variable types**
  Under this tab, you can select the variables.
- **FFB Types**
  Under this tab, you can select FFBs.

## Tab All Types tab

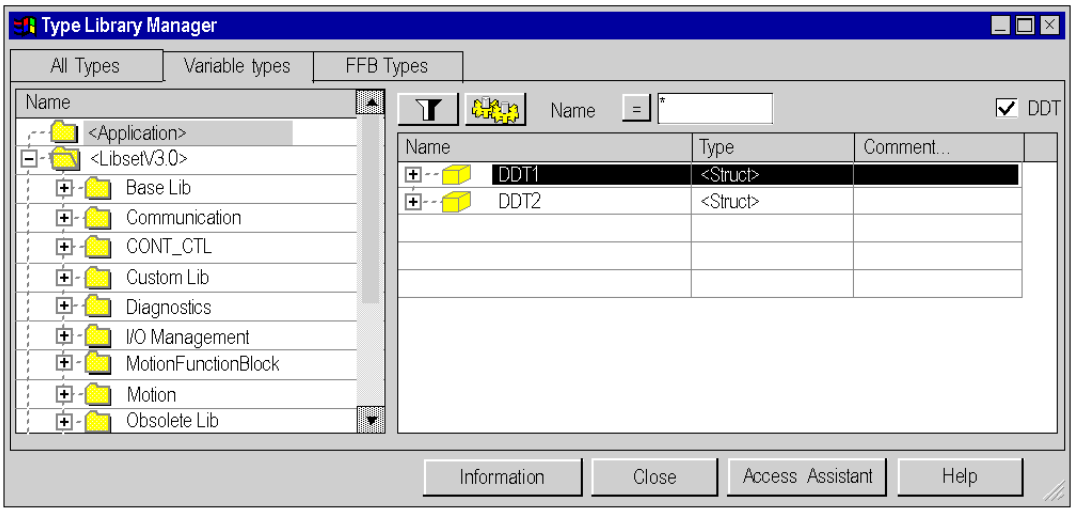


Elements of the **All Types** tab:

| Element | Description |
|---|---|
| Button  | Click this button to update the display corresponding to the filter term defined in the **Name** field. |
| Button  | Click this button to open a dialog box *(see page 412)* for defining the filters. |
| Button  | Click this button to invert the filter. The button changes from **=** to **<>** and vice versa. |
| **Name** (Text Box) | The name of the object to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **EF** | When you enable this check box, Elementary Function types (EFs) are displayed. |
| **EFB** | When you enable this checkbox, Elementary Function Block types (EFBs) are displayed. |
| **DFB** | When you enable this checkbox, Derived Function Blocks (DFBs) are displayed. |
| **DDT** | When you enable this checkbox, multi-element variables (derived variables) are displayed. |
| **Device DDT** | When you enable this checkbox, multi-element variables (derived variables) are displayed. |
| **Name** (Column) | This column displays the names of the function types and function block types and their formal parameters (inputs, outputs). |
| **Type** | This column displays the types of the function types and function block types and the data types of their formal parameters (inputs, outputs). |
| **Comment** | This column displays comments about the function types and function block types and their formal parameters (inputs, outputs). |
| For information on the buttons at the bottom of the dialog please refer to *Areas of the Type Library Manager, page 299*. | |

**NOTE:** Help on Type:
Hold down the SHIFT+F1 keys and then click on the function block type for which you wish to access Help.

## Variable Types tab

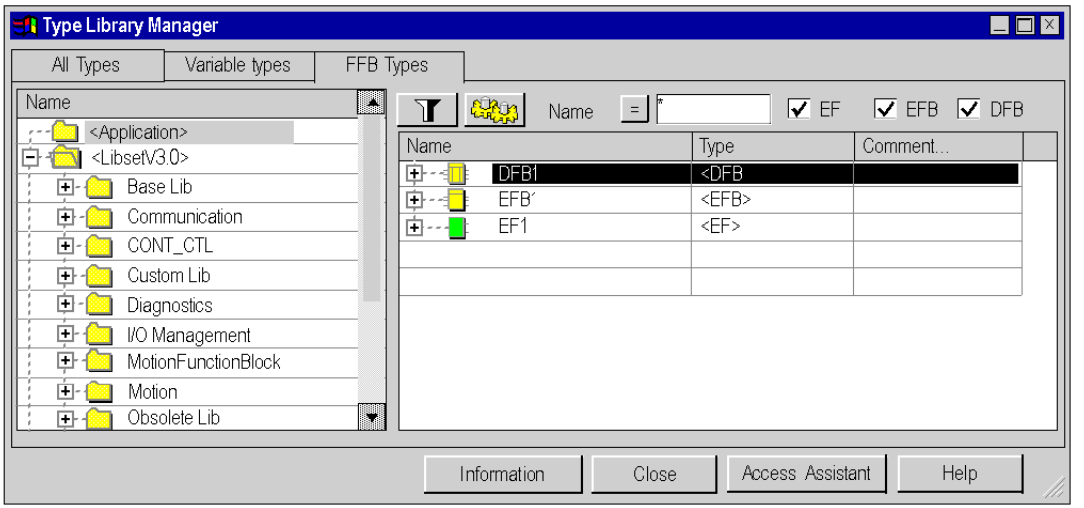



Elements of the **Variable Types** tab:

| Element | Description |
|---|---|
| Button  | Click this button to update the display corresponding to the filter term defined in the **Name** field. |
| Button  | Click this button to open a dialog box *(see page 412)* for defining the filters. |
| Button  | Click this button to invert the filter. The button changes from **=** to **<>** and vice versa. |
| **Name** (Text Box)) | The name of the object to search for can be entered in this text box. You can also use the joker * and **?**. |
| **DDT** | When you enable this checkbox, multi-element variables (derived variables) are displayed. |
| **Device DDT** | When you enable this checkbox, multi-element variables (derived variables) are displayed. |

| Element | Description |
|---|---|
| **Name** (Column) | This column displays the names of the function types and function block types and their formal parameters (inputs, outputs). |
| **Type** | This column displays the types of the function types and function block types and the data types of their formal parameters (inputs, outputs). |
| **Comment** | This column displays comments about the function types and function block types and their formal parameters (inputs, outputs). |
| For information on the buttons at the bottom of the dialog please refer to *Areas of the Type Library Manager, page 299*. | |

## Tab FFB Types




Elements of the **FFB Types** tab:

| Element | Description |
|---|---|
| Button  | Click this button to update the display corresponding to the filter term defined in the **Name** field. |
| Button  | Click this button to open a dialog box *(see page 412)* for defining the filters. |

| Element | Description |
|---|---|
| Button  | Click this button to invert the filter. The button changes from **=** to **<>** and vice versa. |
| **Name** (Text Box) | The name of the object to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **EF** | When you enable this check box, Elementary Function types (EFs) are displayed. |
| **EFB** | When you enable this checkbox, Elementary Function Block types (EFBs) are displayed. |
| **DFB** | When you enable this checkbox, Derived Function Blocks (DFBs) are displayed. |
| **Name** (Column) | This column displays the names of the function types and function block types and their formal parameters (inputs, outputs). |
| **Type** | This column displays the types of the function types and function block types and the data types of their formal parameters (inputs, outputs). |
| **Comment** | This column displays comments about the function types and function block types and their formal parameters (inputs, outputs). |
| For information on the buttons at the bottom of the dialog please refer to *Areas of the Type Library Manager, page 299*. | |

**NOTE:** Help on Type:
Hold down the SHIFT+F1 keys and then click on the function block type for which you wish to access Help.

## Shortcut menu structure

Additional commands can be executed via the shortcut menu of the Type Library Manager.

Shortcut menu commands:

| Menu command | Description |
|---|---|
| **Delete** | Use this menu command to delete a selected user-defined library. |
| **Expand All** | Use this menu command to show the directory structure of a selected library. The same function is also available via the **+** symbol. |
| **Collapse** | Use this menu command to hide the directory structure of a selected library. The same function is also available via the **-** symbol. |
| **Create New Library** | Use this menu command to create a new user-defined library. Compare also *Creating a new library and family, page 312*. |

| Menu command | Description |
|---|---|
| **Rename** | Use this menu command to rename a user-defined library. |
| **Local Print** | Use this menu command to print a list of the FFBs of a selected library. |
| **Display Get Assistant** | Use this menu command to open a window to copy objects into a project. (*Loading an object from the library into the project, page 307*). |
| **Customize Columns...** | Using this menu command you can:<br>● define the columns *(see page 1231)* to be displayed and<br>● define the sequence *(see page 1231)* for the columns shown. |
| **Purge unused Types** | Use the menu command to delete all unused FFBs from the project.<br>**NOTE:** When a FFB is not available for a kind of PLC (after a section import for instance) but still in the application (removed from the section but still in the list of the Application FFBs), the purge feature should be used before building the application. |
| **Compare Project with Library** | This menu command enables you to compare all FFB types of a project with the FFB types of the libraries and update them, if necessary. Compare to *Library Version Management, page 320* |

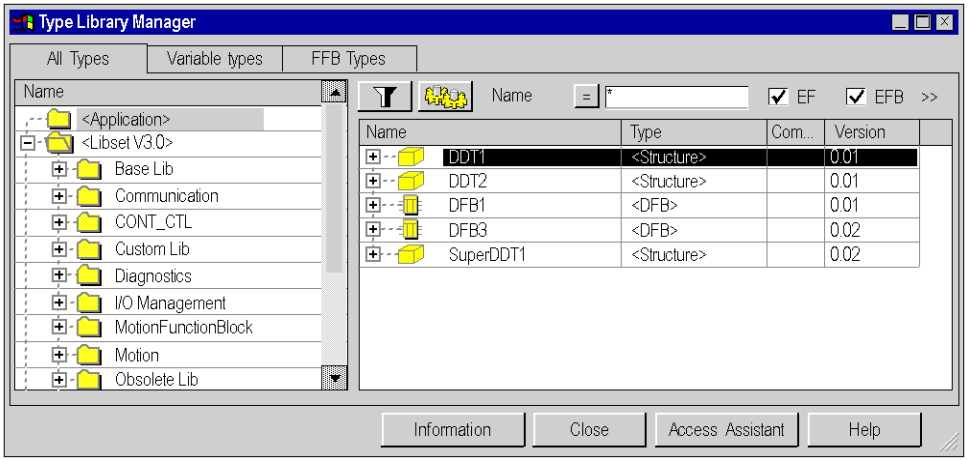
# Loading an object from the library into the project

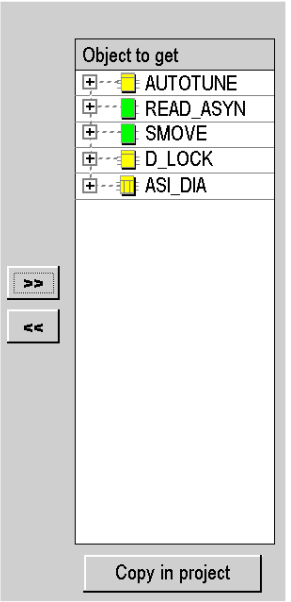
## At a Glance

Loading objects from the library into the projects makes the objects used in a project available quicker without having to run the list of all other objects in the library.

Loading is automatically executed during the first use of a Logic section. You can also load the objects manually from the Type Library Manager.

## Loading an object into the project

| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Manager** option.<br>**Result**: The following window appears.<br>  |

| Step | Action |
|---|---|
| 2 | Click on **Access Assistant** to open the Help window for loading.<br>**Result**: The following area is added on the right of the previously displayed window:<br>  |
| 3 | Use this area to create a list of objects to be loaded by selecting them in sequence beginning in the two sub-windows of Step 1 and then transfer the objects by clicking on the button (double arrows). |
| 4 | Click on **Copy to project** to execute the loading process.<br>**Result**: The objects are now loaded into the project and will be displayed in the tree structure for the project. |

## Loading requirements

| If the name of the object in the project ... | then... |
|---|---|
| does not exist, | the object will be transferred automatically. |
| exists, but the types are different. | **the transfer will be stopped and not performed, and a message listing the causes will be displayed.**<br>**Note**: To load this object you must rename it and restart the loading process. |
| exists and the types are identical. | loading will be executed after you confirm.<br>**Note**: The Consistency Check Assistant *(see page 317)* will ask you to confirm before loading. |

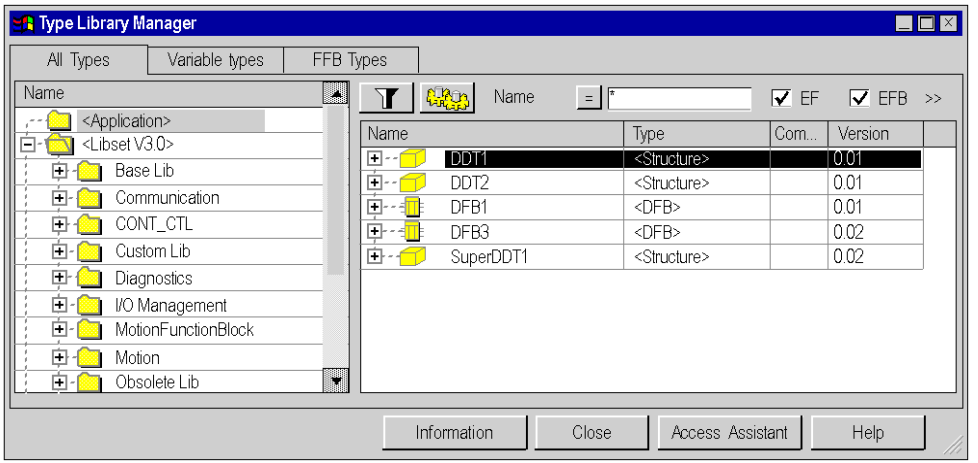
# Transferring an object from the project into the library

## At a Glance

The transfer of an object from the project into the library enables the use of the objects of a project with another project without having to import or export between applications.

Objects transferred to the library can be accessed from all projects.

## Transferring an object to the library

| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Manager** option.<br>**Result**: The following window appears.<br>  |
| 2 | Select the object of the project to be transferred. |
| 3 | Click the right mouse button to open the context menu and select the **Copy to Library** option. |
| 4 | Select the library and target family and click on **OK** to confirm your selection.<br>**Result**: The transfer will be performed automatically.<br>**Note**: If the transfer is not performed automatically, a help window will be displayed to help you correct the problem. |

## Transfer requirements from

| If the name of the object in the library ... | then... |
|---|---|
| does not exist, | the object will be transferred if you have specified the family and the library to which the object is to be transferred. |
| exists, but the types are different. | **the transfer will be stopped and not performed, and a message listing the causes will be displayed.**<br>**Note**: To load this object you must rename it and restart the loading process. |
| exists and the objects are identical. | no action is required because the objects are identical. |
| exists and only the codes of the objects are different. | the transfer will only be executed if the version of the object to be transferred matches or is higher than the version of the object that currently exists in the library.<br>**Note**: The Consistency Check Assistant *(see page 317)* will ask for a confirmation before executing the transfer. |
| exists and only the prototypes of the objects are different. | **the transfer will be stopped and not performed, and a message listing the causes will be displayed.**<br>**Note**: To load this object you must rename it and restart the loading process. |

# Creating a new library and family

## At a Glance

You can create a user-defined library. It contains objects that are available for the development of automation projects, such as:

- Derived Data Types (DDT),
- Function Block Types (EFB/DFB),
- Elementary Functions (EF).

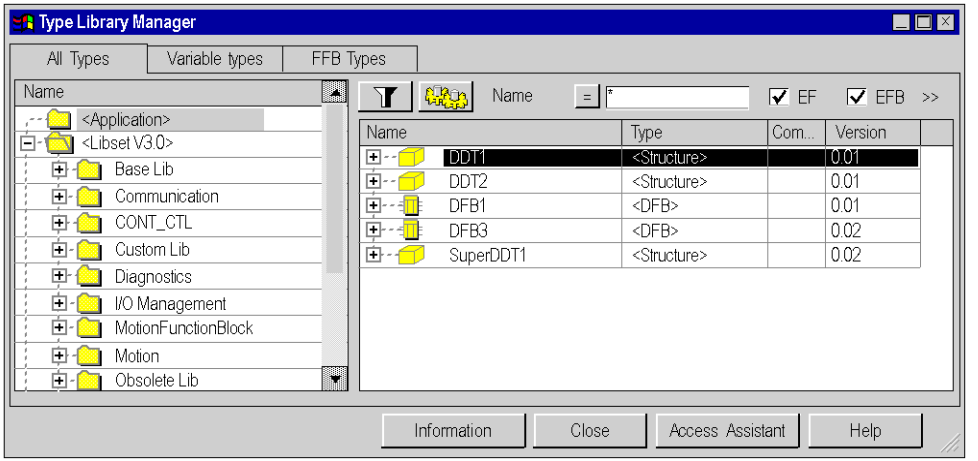
The library is linked to a programming device. This library can access all projects created with this programming device. You can save the Family of the Library *(see page 314)* to reuse *(see page 315)* it on another programming device.

The following steps are required:

- Creating a library,
- Creating one or more families in the new library,
- Linking objects (EF, EFB, DFB, DDT) to each family.

Unity Pro provides a **User-defined Library**. This library contains the family **custom FAMILY**; however, it does not contain objects at this time.

## Create a new library

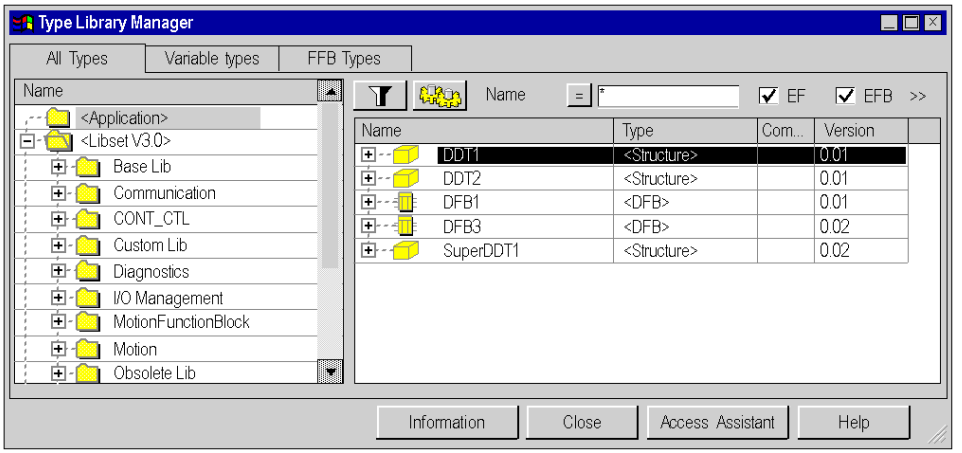
| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Manager** option.<br>**Result**: The following window will be displayed.<br>  |

| Step | Action |
|---|---|
| 2 | Under **Name** select the **Libset** directory. |
| 3 | Select the **Create New Library** command in the context menu. |
| 4 | Enter the name and confirm with **Enter**. |

## Create a new family

| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Manager** option. |
| 2 | Under **Name** select **User-defined Library** or a self-created library. |
| 3 | Select the **Create New Family** command in the context menu. |
| 4 | Enter the name and confirm with **Enter**. |

## Link the object to the family

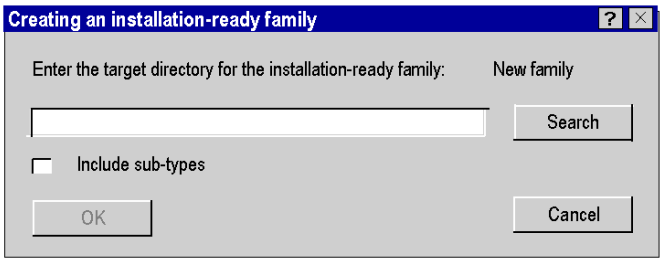
| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Manager** option.<br>**Result**: The following window will be displayed.<br>  |
| 2 | Under **Name** select **User-defined Library** or a self-created library. |
| 3 | In the user-defined library, select **Custom FAMILY** or a self-created family. |
| 4 | Select the object on the right side in the **Name** area. |
| 5 | Drag the object onto **Custom FAMILY** or a self-created family.<br>**Result**: The selected object is located in **Custom FAMILY** or the new family. |

# Creating an installable family

## Introduction

The user-defined library can be saved in a file. It can then be imported into another programming device.

## Procedure

| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Manager** option. |
| 2 | In the **Name** area, display the contents of the **Libset** directory. |
| 3 | Select a family in the **User-defined Library** or a self-created library. |
| 4 | Select the **Create Install-Ready Family**command from the shortcut menu.<br>**Result**: The following window will be displayed.<br>  <br>**Note**: Enable the **Include Sub-Types** checkbox when the family contains sub-types that are not defined in the new project. |
| 5 | Select the directory where the family is to be stored (installed).<br>Use **Search** to select the directory. |
| 6 | Click **OK**. |

## Notes for EF/EFBs

**NOTE:** When you try to create an installable family from a family that contains EF/EFBs, a warning will be displayed that the EF/EFBs will not be exported. For more information about using families with EF/EFBs in other Unity Pro installations, refer to *Updating a family of a library, page 315*.

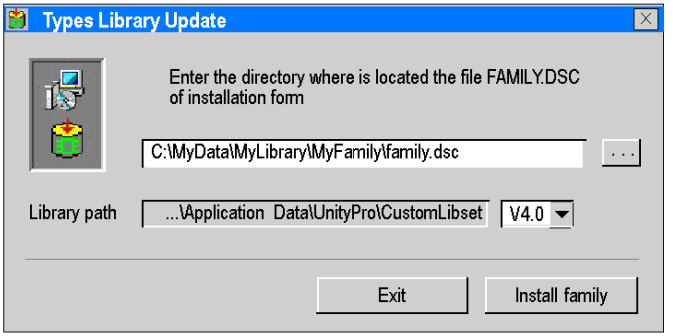
# Updating a family of a library

## Introduction

You can update the type library on a programming device. You can add new EF, EFB, DFB, DDT to use in your program.

**NOTE:** Close Unity Pro before updating.

## Procedure

| Step | Action |
|---|---|
| 1 | Select **Start** → **Programs** → **Schneider Electric** → **Unity Pro** → **Type Library Update**.<br>The **Type Library Update** window will be displayed.<br>  |
| 2 | Select the directory where the family is stored using the **...** button. |
| 3 | Select the **Libset** version (e.g. V4.0) you want to update.<br>(To change the **Library path** please refer to Library configuration options *(see page 557)*.) |
| 4 | Click on **Install Family**. |
| 5 | Answer **Yes** to the displayed question. |
| 6 | Answer **OK** to the displayed warning and click on **Exit**. |

**NOTE:** If you want to use a family in several **Libset** versions of Unity Pro, you must proceed as described above for each **Libset** version.

### Family with EF/EFBs

You have created a user defined family (e.g. toolkit family) with the EFB tool kit and installed it in Unity Pro.

You have also copied DFBs and/or DDTs into the family in Unity Pro.

If you now want to create an installable family, the EF/EFBs from this family will **not** be exported.

### Using families with EF/EFBs in other Unity Pro installations

If you want to use user-defined families with EF/EFBs in other Unity Pro installations, proceed as follows:

| Step | Action |
|---|---|
| 1 | Use your user-defined family (e.g. ToolKit family) to create an **installable family** (*Creating an installable family, page 314*).<br>**Result:** The DFB/DDTs of the family will be exported, the EF/EFBs will **not** be exported. A warning will advise you. |
| 2 | Install the user-defined family with the EF/EFBs you originally created with the EFB toolkit in the new Unity Pro installation. |
| 3 | Install the family you exported in Step 1via **Start → Programs → Schneider Electric → Unity Pro → Type Library Update**.<br>**Result:** Since the original family from the EFB toolkit and the **installable family** from Unity Pro have the same name, a message will appear that this family already exists and you will be asked if you want to replace it. |
| 4 | Answer **No** to the displayed question.<br>**Result:** Another question is displayed whether you want to add the additional types to the existing family instead. |
| 5 | Answer **Yes** to the displayed question.<br>**Result:** All other types (DFB/DDTs) will be added to the family. No existing types will be replaced. |

**NOTE:** If you want to use a family in several **Libset** versions of Unity Pro, you must proceed as described above for each **Libset** version.

# Consistency Check Assistant

## At a Glance

An Assistant will automatically be displayed for complex or multiple operations to check the library or any objects of the projects for consistency to enable easy executing of required processes.

When nested objects are loaded or transferred the Assistant will help you select the objects to be transferred. It will show you the available operations, and which operations do or do not require preparation (renaming, updating, etc.).

**NOTE:** Depending on the complexity of your action, the Assistant will present the objects in a flat or hierarchical format. In some cases a partial list of objects may be selected for processing, and other cases require confirming or canceling the entire operation. The following sections list some examples and meanings.

**NOTE:** The illustrations below show the windows with different columns. This results from the option to configure the columns differently, just like in the Data Editor *(see page 406)*.

## Unchecked elements

When using the Type Library Manager, some elements are not considered during the compatibility check of DFB and EFB types.

These elements are:

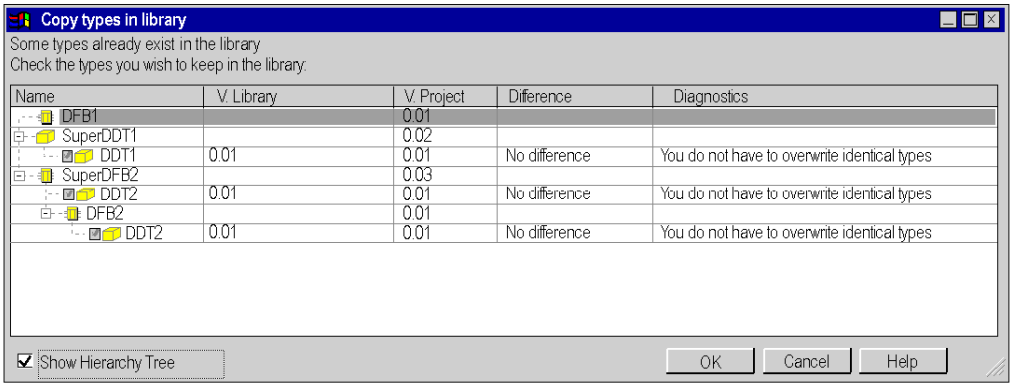
- Initial value elements that contain the initial value of data.
- The Save option showing whether a data element accepts or doesn't accept the backup / restore mechanism.
- The RW program option showing whether a public variable can or cannot be written from the DFB and EFB program.
- The RW program instance option showing whether a public variable can or cannot be written from the program launcher.
- The Pin Number of the parameter. (**Note**: This check is only not considered with DFBs)

## Libset Version 2.2

**NOTE:** Upgrading the **Libset** version of the project may lead to have a list of differences during the comparison of EFs, EFBs, DFBs used in the application with the library. In case of a project build with a **Libset** version 2.2, the version number (V1.0) may be identical with the selected **Libset**, even if differences are existing. Upgrading the project with the new **Libset** requires a rebuild of the project and a download in the PLC.

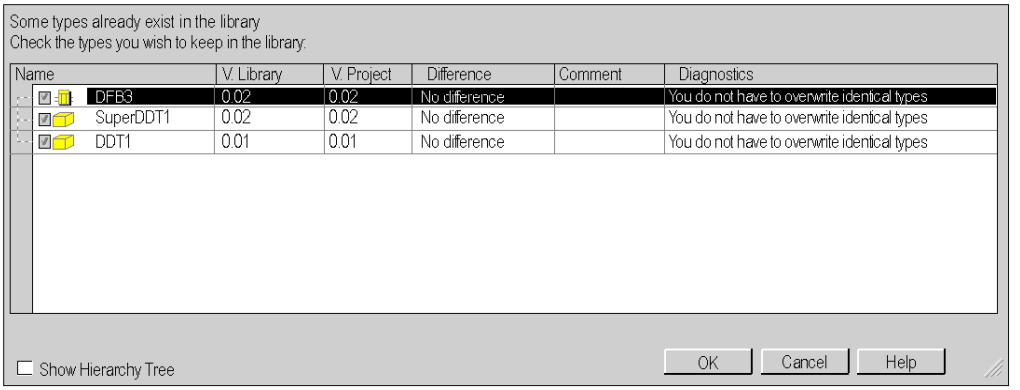
## Example of a hierarchal illustration with global confirmation.

The following illustration shows an example of the assistant for a hierarchal view with required global confirmation. The window shows that certain object types already exist in the library and you are asked to confirm whether the types should remain in the library.



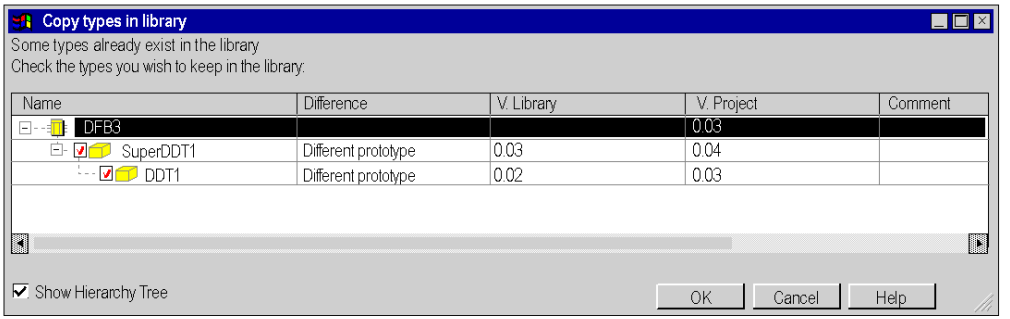

## Example of a flat illustration with global confirmation.

The following illustration shows an example of the assistant for a flat (non-hierarchal) view with required global confirmation. The window shows that certain object types already exist in the library and you are asked to confirm whether the types should remain in the library.



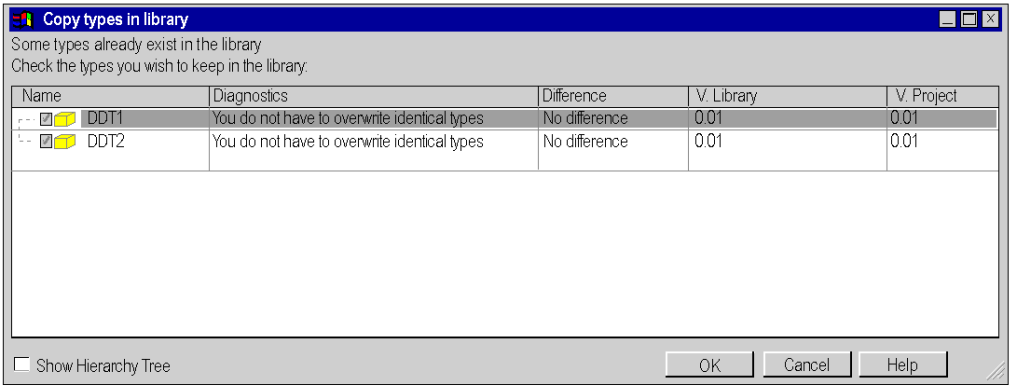
### Example of a hierarchal illustration with selective confirmation.

The following illustration shows an example of the assistant for a hierarchal view with a selective confirmation option. The window shows that certain object types already exist in the library and you are asked to confirm whether certain types should remain in the library.




### Example of a flat illustration with selective confirmation.

The following illustration shows an example of the assistant for a flat (non-hierarchal) view with a selective confirmation option. The window shows that certain object types already exist in the library and you are asked to confirm whether certain types should remain in the library.



# Library Version Management

## At a Glance

The Type Library Manager enables you to compare all FFB types of a project with the FFB types of the libraries.

If there are differences between the FFB types in the project and FFB types in the library, the **LIBRARY Version Management** dialog will be displayed showing the relevant information.

This dialog box has two buttons to updated the FFB types of the project with the FFB types of the libraries.

## Compare project to library

The **Compare Project to Library command...** Begin via the context menu of the Type Library Manager (right-click).

## LIBRARY Version Management tab.

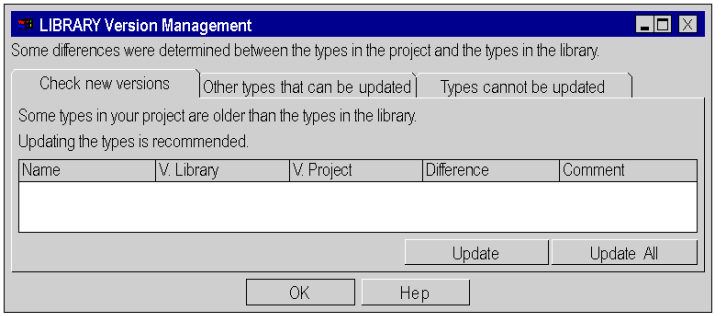
The **LIBRARY Version Management** consists of three tabs:

- **Check new versions**
- **Other types that can be updated**
- **Types cannot be updated**

The first two tabs have buttons to update the FFB types of the project with the FFB types of the libraries.

## Register: Check new versions

This tab displays FFB types of the project that are older than the types of the libraries.



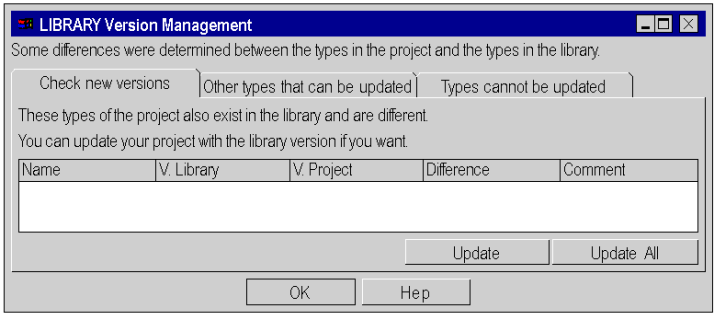
Elements of the **Check New Versions** tab:

| Element | Description |
|---|---|
| **Name** | Name of the FFB type |
| **V. Library** | Version of the FFB type in the library |
| **V. Project** | Version of the FFB type in the project |
| **Difference** | Determined difference between the two versions |
| **Comment** | Comment about the version difference |
| Button<br>**Update** | Updates the selected FFB type with the FFB type from the library. |
| Button<br>**Update All** | Updates all FFB types shown in this tab with the FFB types from the library. |
| Button<br>**OK** | Closes the window. |
| Button<br>**Help** | Opens Unity Pro Help |

**NOTE:** Updating FFB types leads to **Rebuild All Project**. The PLC must be stopped and a global download of the project must be performed.

## Register: Other types that can be updated

This tab shows FFB types of the project that are different from those in the libraries, but can be updated.



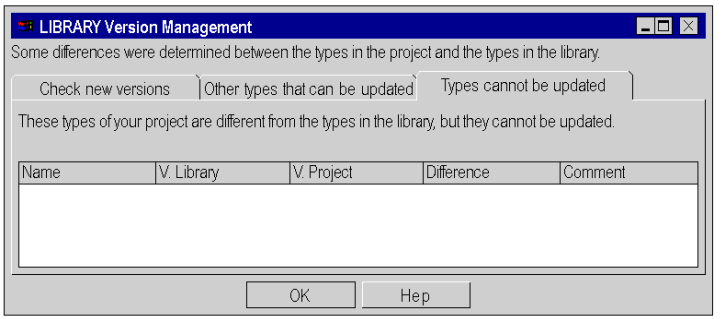
Elements of the **Other types that can be updated** tab:

| Element | Description |
|---|---|
| **Name** | Name of the FFB type |
| **V. Library** | Version of the FFB type in the library |
| **V. Project** | Version of the FFB type in the project |
| **Difference** | Determined difference between the two versions |
| **Comment** | Comment about the version difference |
| Button<br>**Update** | Updates the selected FFB type with the FFB type from the library. |
| Button<br>**Update All** | Updates all FFB types shown in this tab with the FFB types from the library. |
| Button<br>**OK** | Closes the window. |
| Button<br>**Help** | Opens Unity Pro Help |

**NOTE:** Updating FFB types leads to **Rebuild All Project**. The PLC must be stopped and a global download of the project must be performed.

## Register: Types cannot be updated

This tab shows FFB types of the project with differences from those in the libraries that are so great that they cannot be updated. In this case you could updated FFB types only after you did an export/import of your project.



Elements of the **Types that cannot be updated** tab:

| Element | Description |
|---|---|
| **Name** | Name of the FFB type |
| **V. Library** | Version of the FFB type in the library |
| **V. Project** | Version of the FFB type in the project |
| **Difference** | Determined difference between the two versions |
| **Comment** | Comment about the version difference |
| Button<br>**OK** | Closes the window. |
| Button<br>**Help** | Opens Unity Pro Help |

# Using the type library of an older Unity Pro Version

## At a Glance

If you have already used an older version of Unity Pro, and if you have created your own type library and families to store DFBs or DDTs, you have to provide this own type library and family for the new version of Unity Pro.

## Providing the type library and families

Follow the steps below to install a new version of Unity Pro:

| Step | Action |
|---|---|
| 1 | Start the Unity Pro software. |
| 2 | Use **Tools** to open the **Type Library Manager**. |
| 3 | Create a ready-to-install family for each family containing your own DFBs or DDTs (via the context menu for the applicable family). Save these families on your hard drive (but not in the Schneider Electric directories). |
| 4 | Install the new version of Unity Pro. |
| 5 | Go to **Programs** → **Schneider Electric** → **Unity Pro** and start the Tool **Type Library Update** and install all of your families.<br>**Note:** If you want to use your families in several **Libset** versions, you must repeat this step for every **Libset** version. |
| 6 | Start the Unity Pro software. |
| 7 | Make sure that all of your families have been installed. |
| 8 | Create an application. |
| 9 | Copy each DFB that belongs to your own type library from the library into your application. |
| 10 | Then copy each DFB from the application into the library family that it came from. |
| 11 | All of your libraries and families are now ready for the new Unity Pro Version. |

# Part II
# Data Editor

## Object of this part

It guides the user through the general features available in the data editor in order to:

- Create data types,
- Create data type instances,
- search for data types or instances.

## What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 9 | Description of the Data Editor | 327 |
| 10 | Creating derived data types (DDT) | 333 |
| 11 | Creating data types of user function blocks (DFB) | 357 |
| 12 | Managing of instances of data belonging to the family of function blocks (EF) | 359 |
| 13 | Managing variable instances belonging to the EDT/DDT/IODDT/Device DDT families | 369 |
| 14 | General Functions | 403 |

# Chapter 9
# Description of the Data Editor

## Object of this chapter

This chapter gives an overview of the data editor.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Data Editor Access | 328 |
| Description of Data Editor tabs | 330 |

# Data Editor Access

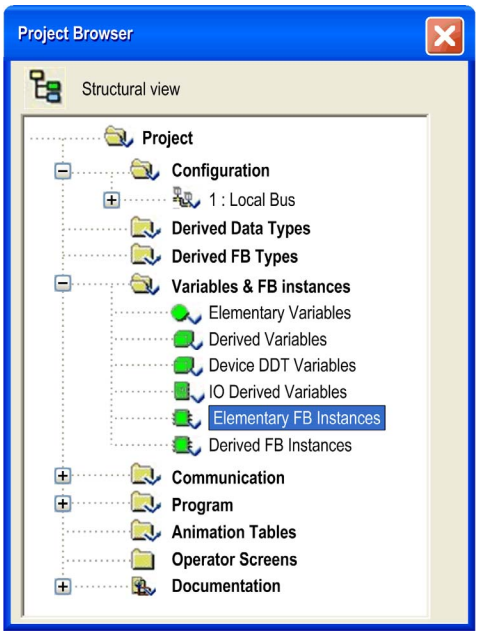
## Introduction

The project can be accessed from the structural view in the data editor. It has features which support the following:

- Creating data types,
- Archiving or using function block data types in a library,
- Creating data type instances,
- Displaying data structures in hierarchical manner,
- Searching/sorting/filtering data.

## Accessing the Data Editor

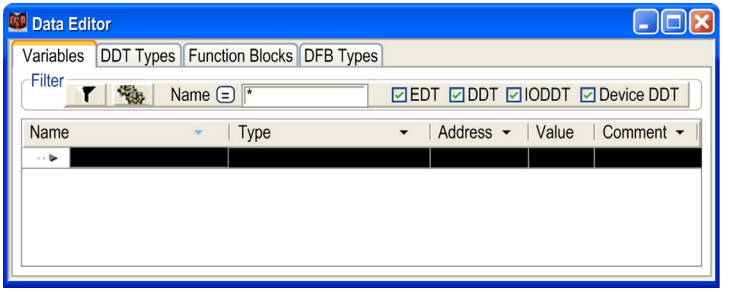
Structural view of a project:



Carry out the following steps:

| Step | Action |
|---|---|
| 1 | With the right mouse-button click on the **Variables and FB instances** directory. |
| 2 | Select the **Open**command in the shortcut menu.<br>**Result**: The data editor opens. The **Variables** tab is shown by default. |

Data Editor:

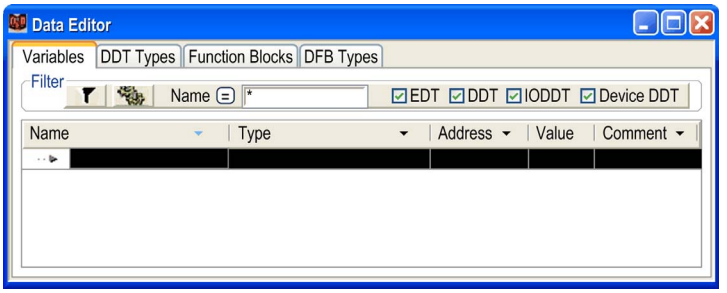



**NOTE:** You can open the editor directly via data instances which belong to a family defined by you. If you want to do this open the editor from one of the **subdirectories** in the **Variables and FB instances** directory. The variables to be displayed will be listed in the selected subdirectory.

# Description of Data Editor tabs

## Data Editor Elements

Appearance:




Description:

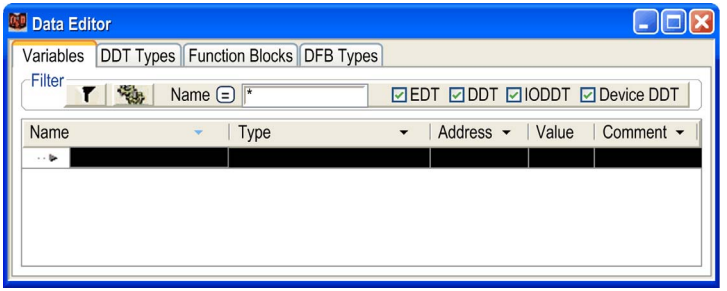
| Element | Description |
|---|---|
| **Filter** | |
| Button  | Clicking on this button updates the display corresponding to the filter term defined in the **name** field. |
| Button  | Clicking on this button opens a dialog box *(see page 412)* for defining the filters. |
| Button  | Clicking on this button inverts the filter. The button changes from **=** to **<>** and vice versa. |
| Name | Column where you enter the name (symbol) of the instance or the data type you want to display. You can use the place holders (* or **? for this.**). |
| EDT | Displaying elementary data types |
| DDT | Displaying derived data types |
| IODDT | Display the derived data types (DDT) that reference inputs/outputs |
| Device DDT | Display the device derived data types |

| Element | Description |
|---|---|
| **Properties** | |
| Name | Column where you enter the name (symbol) of the instance or data type |
| Type | Column where you select the variable type. |
| Address | Column where you reference the variable instance with an address in the controller (located instance). |
| Value | Column where you initialize the variable. |
| Comment | Column where you enter the comment for the variables. |

**NOTE:** The different fields can be assigned parameters *(see page 404)*.

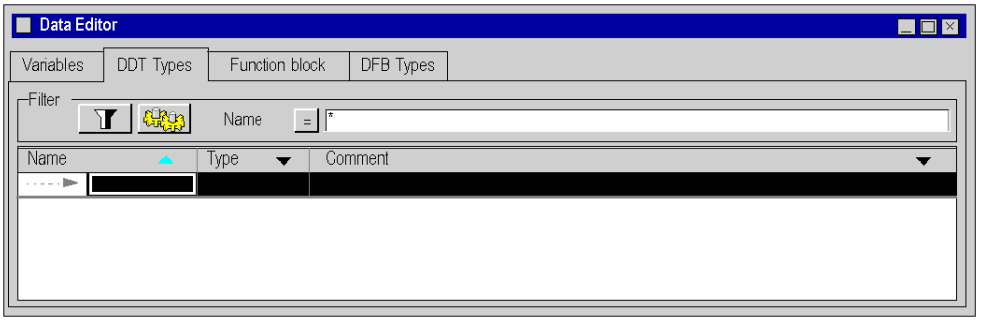
## Tab: Variables

The **Variables** tab enables the administration of variable instances that belong to the families EDT/ DDT/ IODDT/ Device DDT



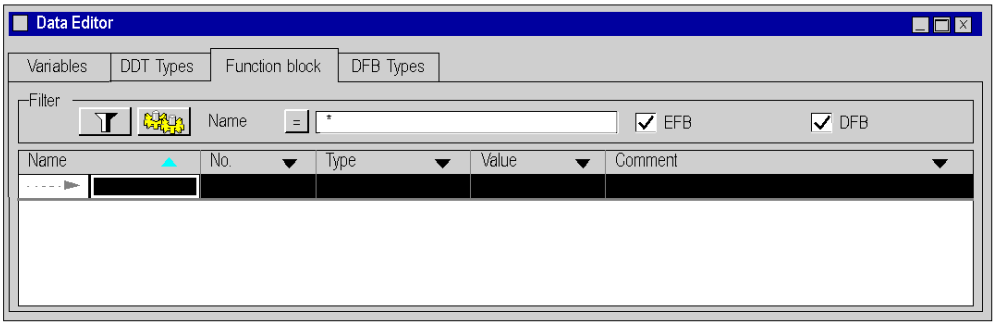

## Tab: DDT Types

The **DDT Types** tab is used to manage derived data types (structures or arrays):



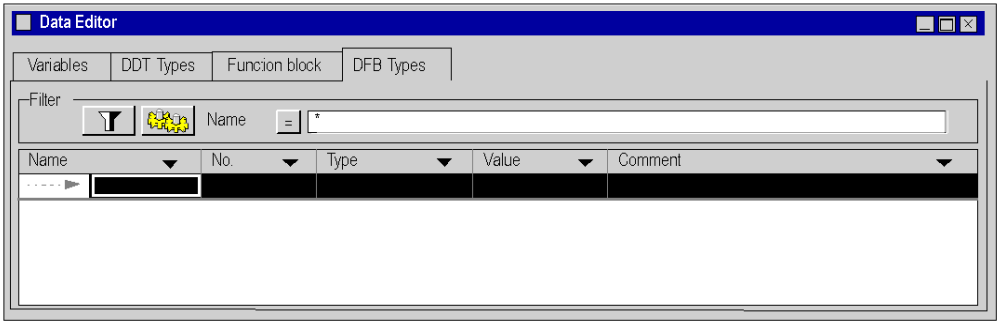
### Tab: Function blocks

The **Function Blocks** tab enables the management of the variable instances of type EFB or DFB that belong to the function block family.




### Tab: DFB Types

The **DFB Types** tab enables the administration of data types of derived function blocks (DFB).



# Chapter 10
# Creating derived data types (DDT)

## Object of this chapter

This chapter guides the user through the creation and management of derived data types (DDT).

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Access to derived data types (DDT) | 334 |
| Creation of a derived data type (DDT) | 335 |
| Administration of derived data types and their elements | 346 |
| Creating Nested Derived Data Types | 348 |
| Save Derived Data Types (DDT) | 355 |

# Access to derived data types (DDT)

## Procedure

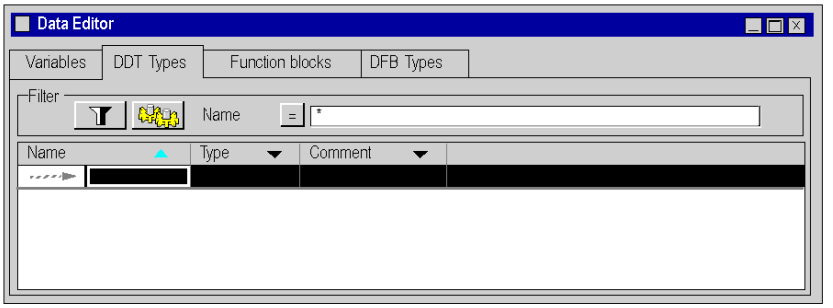
You have the following two options to access the Data Editor to create derived data types:

- from the project structure view via the **Variables and FB Instances** directory,
- from the project structure view via the **Derived Data Types** directory,

### Access via the "Variables and FB Instances" directory.

| Step | Action |
|---|---|
| 1 | In the project structure view, switch to the **Variables and FB Instances** directory. |
| 2 | Select the **Open** command in the shortcut menu and the Data Editor will open. |
| 3 | Select the **DDT Types** tab in the Data Editor. |

Example:




### Access via the "Derived Data Types" directory.

| Step | Action |
|---|---|
| 1 | Switch to the **Derived Data Types** directory in the project structure view. |
| 2 | Select the **Open** command in the shortcut menu, which will open the Data Editor that is automatically shown in the **DDT Types** tab. |

# Creation of a derived data type (DDT)

## Introduction

A derived data type corresponds with:

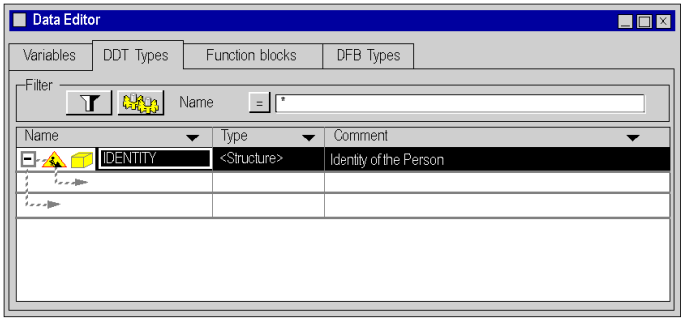
A derived data type is one of the following

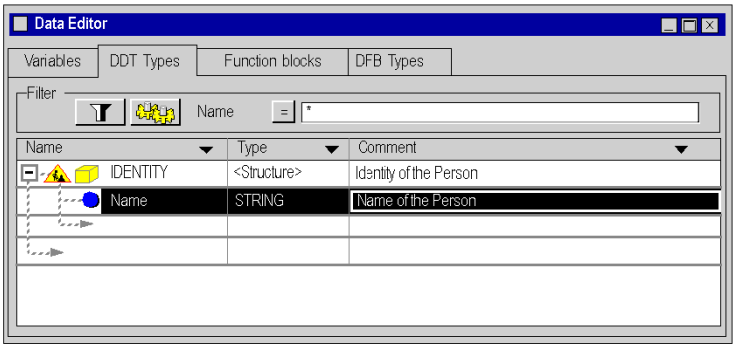
- a structure type:
  - Structure for input/output data, the type is not created by the user in this case, but by the **manufacturer** (IODDT),
  - Structure for other data; in this case the type is **created by user**.
- an array type.
- a reference type.

## Memory Structure

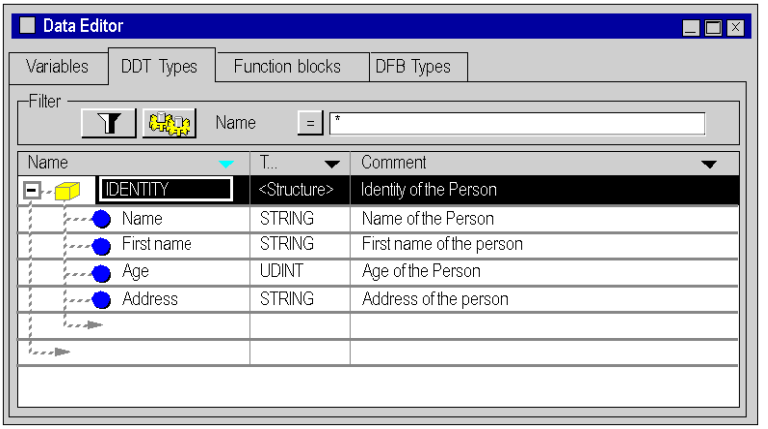
The different PLC families vary in memory structure.

## Creating a Structure Type

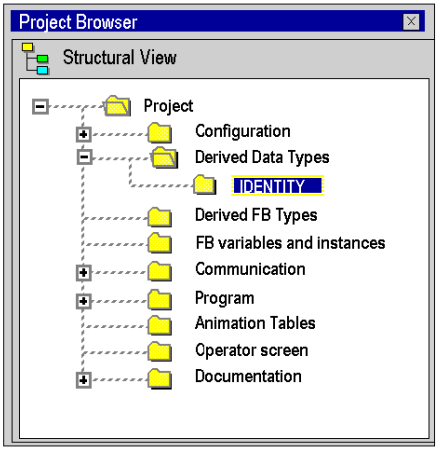
| Step | Action |
|---|---|
| 1 | Double click in the Data Editor in the **DDT Types** tab on the empty field (marked by an arrow **Name**, and enter the name of the structure type (e.g. IDENTITY); the default type is **<Struct>**. Enter a comment (optional). |
| 2 | Call up the newly created structure by clicking on **+**.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Double-click on the corresponding **Name** field and enter the name of the first element of the structure (e.g. Name, Age) and its type and comment (optional). Example:   |
| 4 | Double-click on the next cell (marked with an arrow) to enter the name of the next element etc. |
| 5 | Go to the name of the structure type (`IDENTITY`) and select **Analyze Type** in the shortcut menu. |

**Result**: The structure was successfully analyzed (the symbol in front of the structure type has changed):



**Result**: The IDENTITY structure is created in the structure view of the project:




**NOTE:** Using the shortcut menu and the **Open** command you can access the IDENTITY structure.

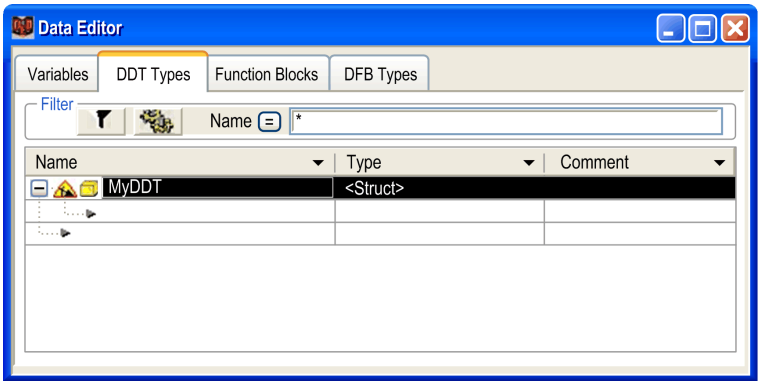
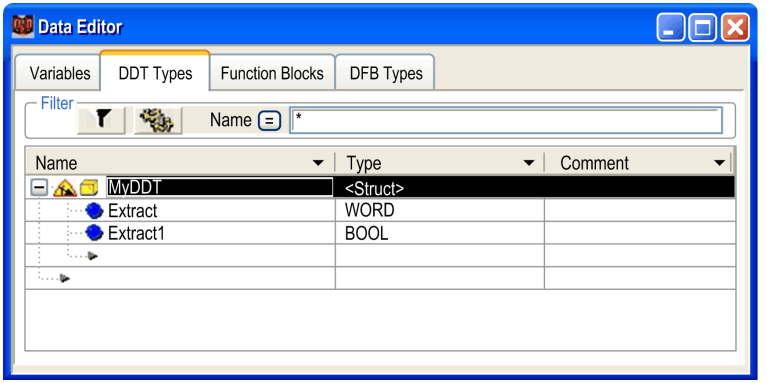
## Creating a Structure Type with an Extracted Bit

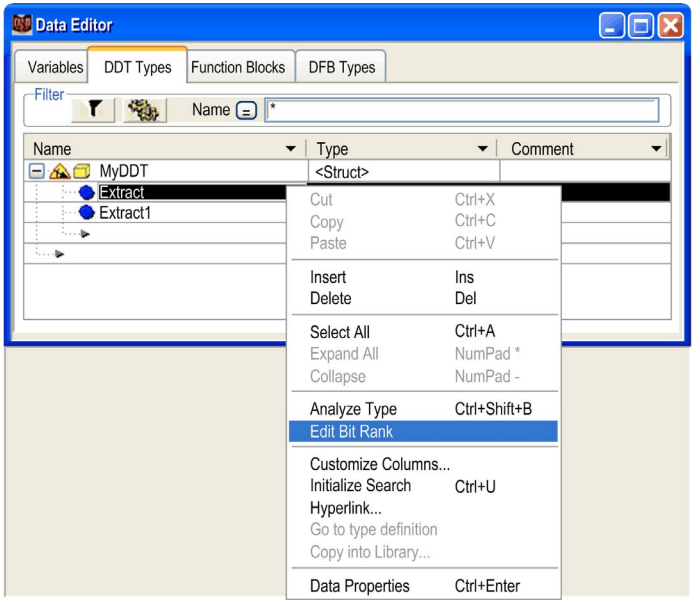
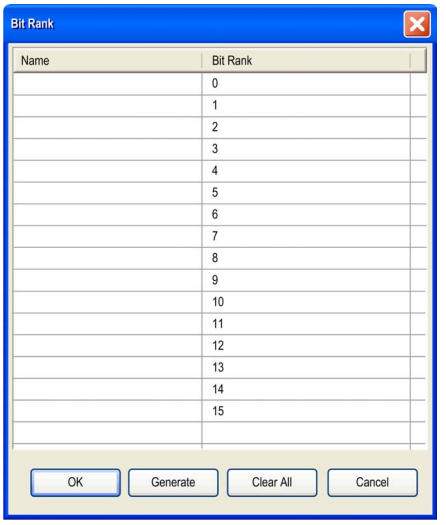
The user can create structure type using extracted bit.

The **Bit Rank** dialog box is accessible by right clicking on the instance or data type which type must be:

- WORD
- UINT
- INT
- BYTE
- an extracted bit with a compatible parent

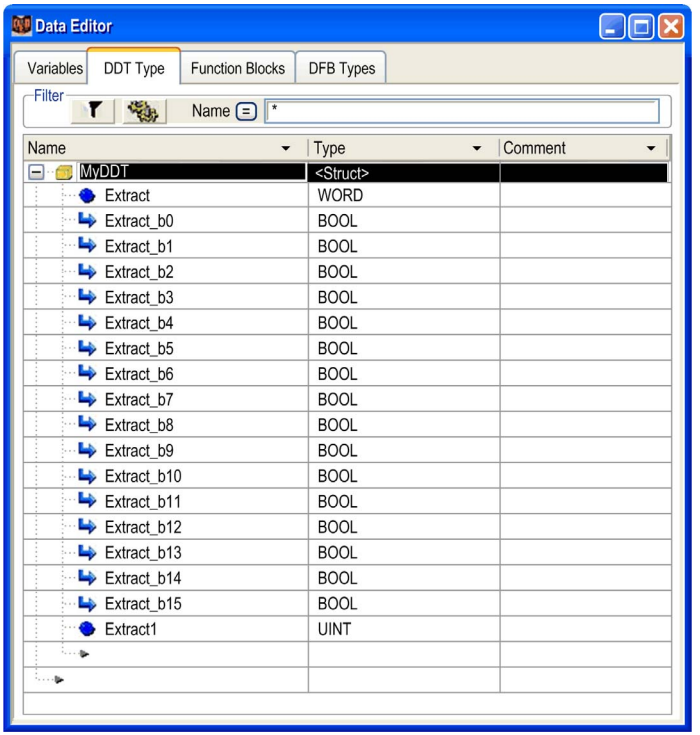
| Step | Action |
|---|---|
| 1 | Double click in the Data Editor in the **DDT Types** tab on the empty field (marked by an arrow **Name**, and enter the name of the structure type (for example, MyDDT); the default type is **<Struct>**. Enter a comment (optional). |

| Step | Action |
|---|---|
| 2 | Call up the newly created structure by clicking on **+**:<br>  |
| 3 | Double-click on the corresponding **Name** field and enter the name of the first element of the structure (e.g. Extract, Extract1) and its type and comment (optional):<br>  |
| 4 | Double-click on the next cell (marked with an arrow) to enter the name of the next element etc. |

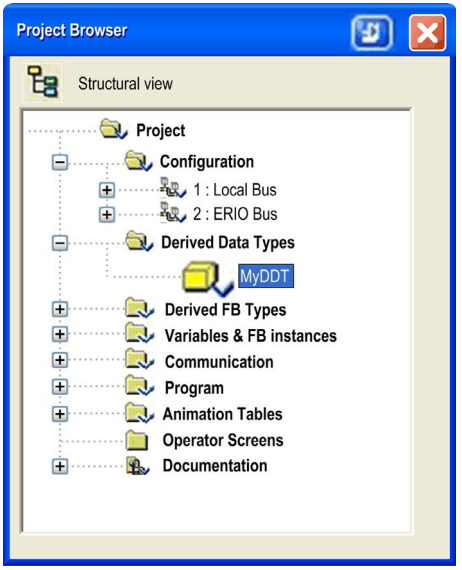
| Step | Action |
|---|---|
| 5 | Select the variable to create the extracted bit and right-click on the variable.<br>The **Edit Bit Rank** is available in the contextual menu list:<br>  <br><br>Here below is the **Bit Rank** dialog box:<br>  <br> |

| Step | Action |
|---|---|
| 6 | Within the **Bit Rank** menu, the user can select to named each variable manually or automatically using the **Generate** button.<br>**NOTE:** The Bit name can be edited manually directly in the **Data Editor**. |
| 7 | Go to the name of the structure type (`MyDDT`) and select **Analyze Type** in the shortcut menu. |

**Result**: The structure was successfully analyzed (the symbol in front of the array type has changed):



**Result**: The `MyDDT` structure is created in the structure view of the project:

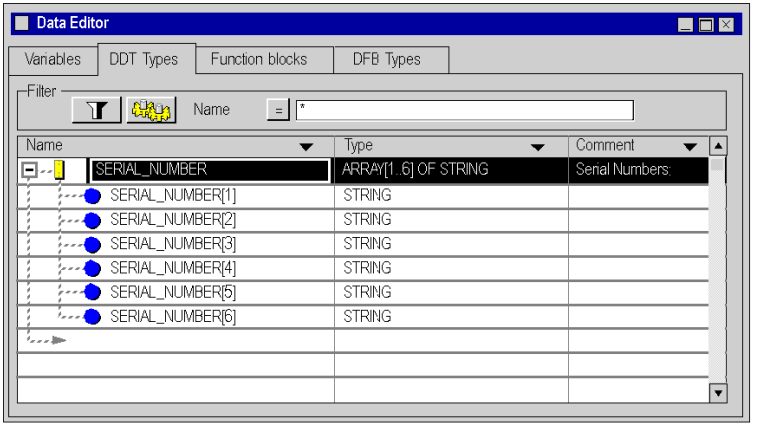



**NOTE:** Using the shortcut menu and the **Open** command you can access the `IDENTITY` structure.

## Creating an Array Type

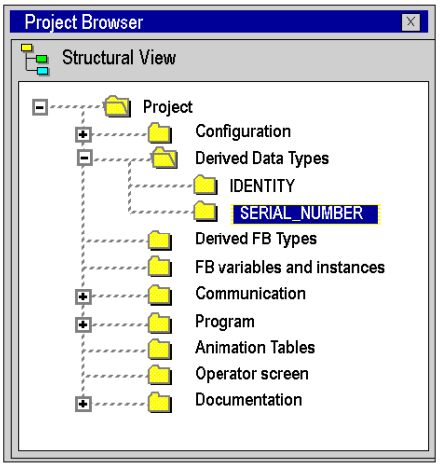
Use this procedure to create an array type:

| Step | Action |
|---|---|
| 1 | Double click in the **Data Editor** in the **DDT Types** tab on the empty field (marked by an arrow **Name**, and enter the name of the array type (e.g. `SERIAL_NUMBER`); the default type is **<Struct>**. Enter a comment (optional). |
| 2 | Double click on the **Type** field, select **<Array>**, and confirm by pressing **Enter**. **Result**: The Variable Type Selection dialog appears. |
| 3 | The **ARRAY** checkbox is automatically checked:<br>● Enter the size of the array in the corresponding field (for example, 1..6)<br>● In the **libraries/families** area select the types that should be displayed:<br>❍ the **<EDT>** directory for an elementary type<br>❍ the **<Application>** directory for a derived type already in the application<br>❍ the **<Libset>** directory for a derived type archived in a library |
| 4 | Confirm with **OK**. |

| Step | Action |
|---|---|
| 5 | Change to the name of the array type and go to the shortcut menu and select **Analyze Type**.<br>**Result**: The array was successfully analyzed (the symbol in front of the array type has changed):<br>  |

**NOTE:** Comments cannot be entered for the sub elements of the array type.

The array type `SERIAL_NUMBERS` is created in the project structure view:




**NOTE:** You can access the array of the `SERIAL_NUMBER` type in the context menu using the **Open** command.

## Creating a Reference Type

| Step | Action |
|---|---|
| 1 | Double click in the **Data Editor** in the **DDT Types** tab on the empty field (marked by an arrow **Name**, and enter the name of the reference type (e.g. `MyRefInt`); the default type is **<Struct>**. Enter a comment (optional). |
| 2 | Double click on the **Type** field, select **<Reference>**, and confirm by pressing **Enter**.<br>**Result**: The Variable Type Selection dialog appears. |
| 3 | The **REF_TO** checkbox is automatically checked. Select the reference data type, in this example: INT. |
| 4 | Confirm with **OK**. |
| 5 | In the `Data Editor`, right-click the name `MyRefInt` and select **Analyze Type**.<br>**Result**: The reference was successfully analyzed (the symbol in front of the reference type has changed). |

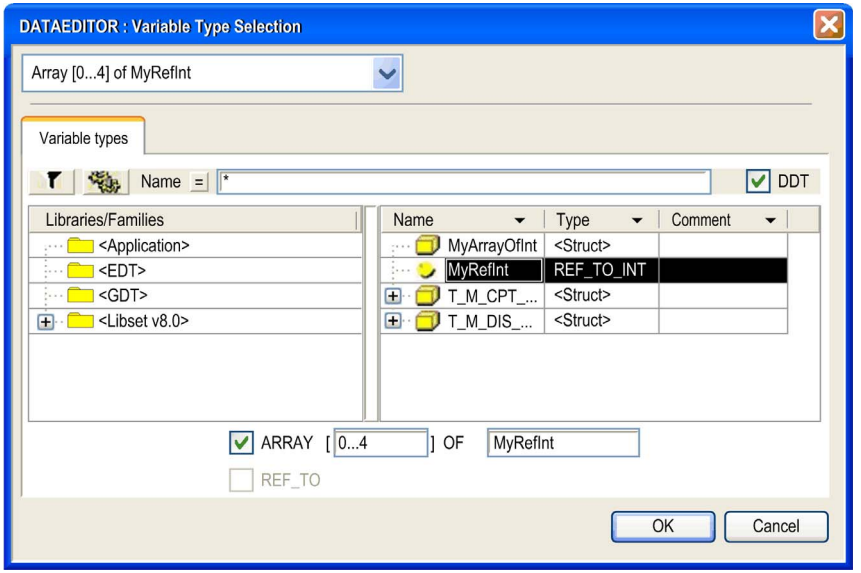
The reference type `MyRefInt` is created in the **Project Browser Structure view**.

**NOTE:** You can access the reference of the `MyRefInt` type using the contextual menu using the **Open** command.

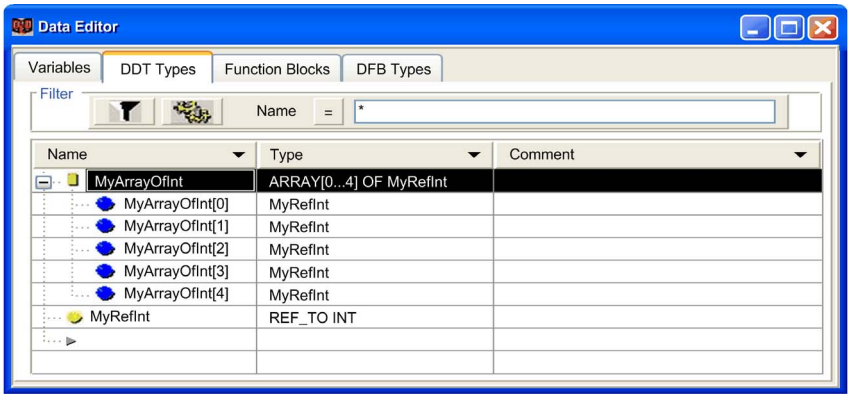
## Array of References

You cannot make directly, for example, an Array of REF_TO INT. You must first make a reference DDT with type REF_TO INT as in the figure above. Then you make a new DDT type, an array of the references.

In **<Application>**, select `MyRefInt`:




After confirming with **OK** and performing an **Analyze Type**, you obtain an array of references:

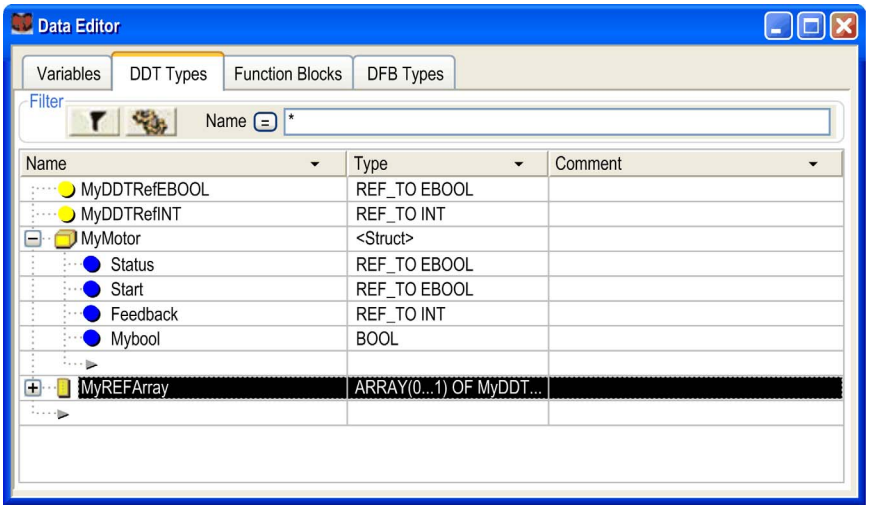


## Structure with References

This example shows the creation of an array of reference type variables:

| Step | Action |
|---|---|
| 1 | Double click **DDT Types** tab in the **Data Editor** on the empty field (marked by an arrow **Name**, enter the name of the structured type and press ENTER. The default type is **<Struct>**. |
| 2 | Double click **Types** and select **<Array>** in the drop-down menu. |
| 3 | Press ENTER to open the **Variable Type Selection** dialog. |
| 4 | Set the size of the array. |
| 5 | Select **<Application>**. |
| 6 | Select `MyDDTRefINT`. |

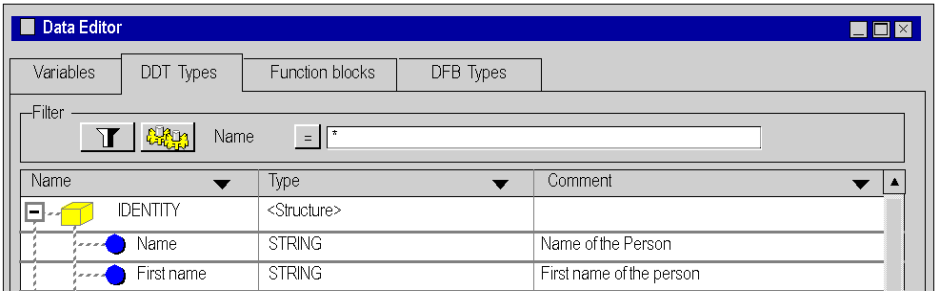
The array of reference types example:



# Administration of derived data types and their elements

## Inserting an element into a derived data type

Execute the following steps:

| Steps | Action |
|---|---|
| 1 | Display the desired data type in the data editor in the **DDT Types** tab. |
| 2 | Place the cursor on the element in front of which you want to insert another element and select the **Insert new** command in the shortcut menu.<br>Inserting an element into an address element<br>  |
| 3 | Enter the attributes of the element into the newly created row. |

## Deleting an element belonging to a derived data type

Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Display the desired data type in the data editor in the **DDT Types** tab from which you wish to delete the element. |
| 2 | Select the desired element. |
| 3 | Use the shortcut menu to select the **Delete** command. |

**NOTE:** Elements belonging to protected data types cannot be deleted.

### Deleting a derived data type:

Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the desired data type in the data editor in the **DDT Types** tab. |
| 2 | Using the context menu select the **Delete** command. |

**NOTE:** Data types that exist as an instance in the application cannot be deleted.

# Creating Nested Derived Data Types

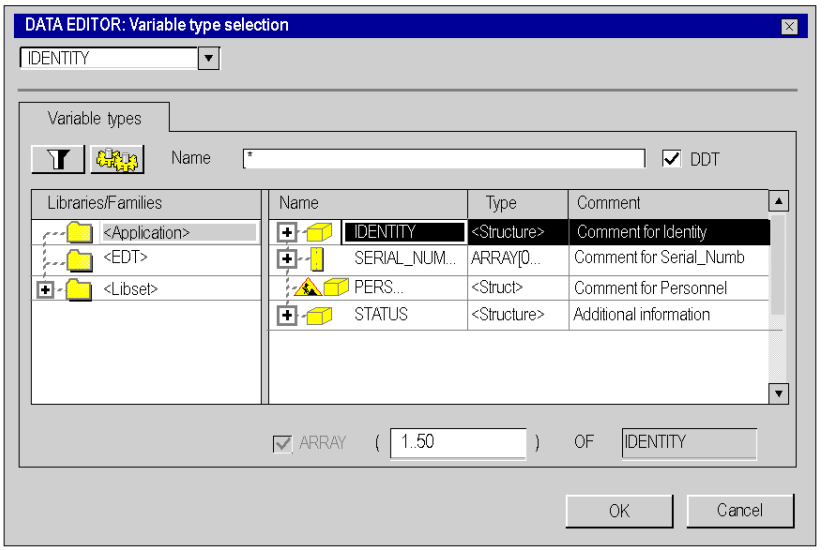
## Introduction

Data types that are part of the DDT family can be nested. The nesting can be organized as follows:

- in structures that contain arrays or structures
- in arrays that contain arrays or structures
- in a combination of these different nestings

## Array, that contains structures

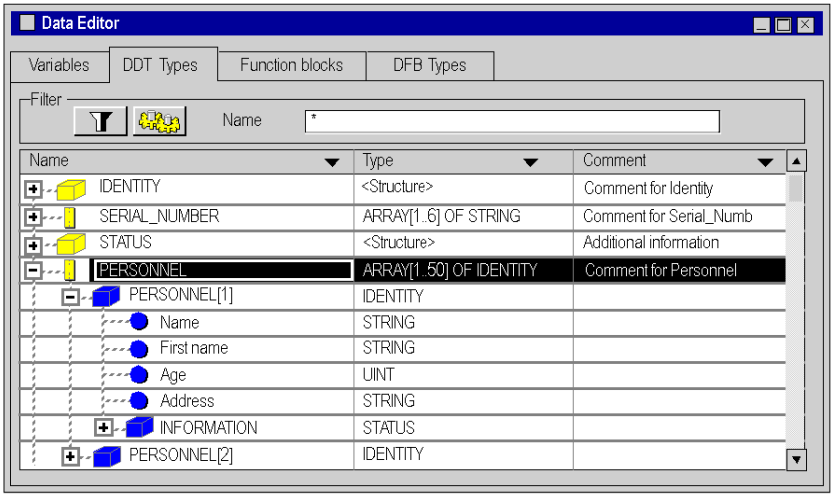
Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Double click in the Data Editor in the **DDT Types** tab on the empty field (marked by an arrow **Name**, and enter the name of the array type (e.g. PERSONNEL); the default type is **<Struct>**. Enter a comment (optional). |
| 2 | Double click on the **Type**field, select **<Array>**, and confirm by pressing **Enter**.<br>**Result:** The **Variable Type Selection** appears. |
| 3 | The **ARRAY** checkbox is automatically checked:<br>● enter the size of the array in the corresponding field<br>● In the **libraries/families** area select the following:<br>❍ the **<EDT>** directory for an elementary type<br>❍ the **<Applications>** directory for a type already in the application<br>❍ the **<Library>** directory for an already archived type<br>See the following window:<br>  |
| 4 | Enable the **DDT** checkbox, which will display the derived data types. |
| 5 | Select the desired structure in the **Name** column and confirm with **OK**. |
| 6 | Change to the name of the array type and go to the context menu and select **Analyze Type**. |

**NOTE:** Comments cannot be entered for the sub elements of the array type.

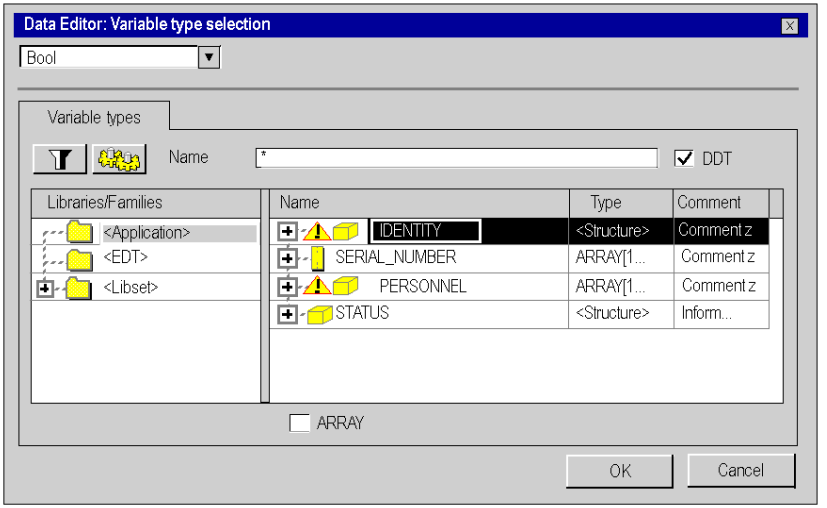
**NOTE:** Changing the boundaries of an anonymous array lead to the lost of the initial values of that array. Those values must be re-entered after the modification.

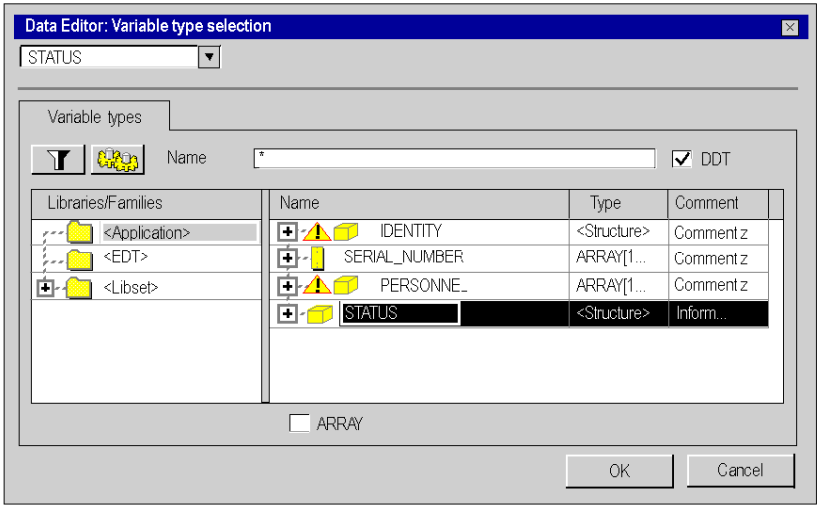
Array of the type **PERSONNEL** that contains structures of the type **IDENTITY**:



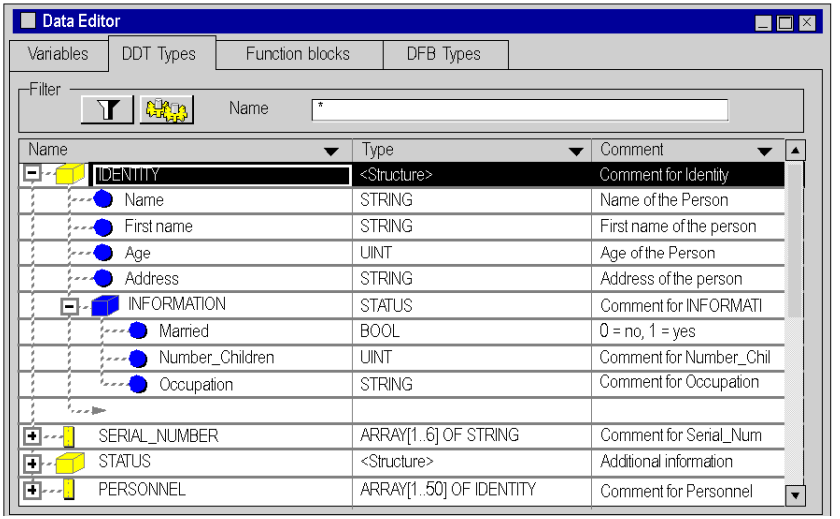
## Structure containing a structure

Execute the following actions (provided a structure exists):

| Step | Action |
|---|---|
| 1 | Click on **+** to display the type of the structure in the **DDT Types** tab in the data editor, to which you want to add an element. |
| 2 | Double click on the empty **Name**field and enter the name of the new structure type element. Enter a comment (optional). |
| 3 | Double-click on the corresponding **Type** field and click on .<br>  <br>The following window appears.<br>  |

| Step | Action |
|---|---|
| 4 | In the **libraries/families** area select:<br>● the **<EDT>** index for an elementary type<br>● the **<Applications>** directory for a type that already exists in the application<br>● the **<Library>** for an already archived type<br>See the following example:<br>  |
| 5 | Enable the **DDT** checkbox, which will display the derived data types. |
| 6 | Select the desired structure in the **Name** column and confirm with **OK**. |
| 7 | Switch to the name of the structure type and select the **Analyze Type**entry in the context menu. |

Example: **IDENTITY** structure that contains an **INFORMATION** structure of the **STATUS** type.

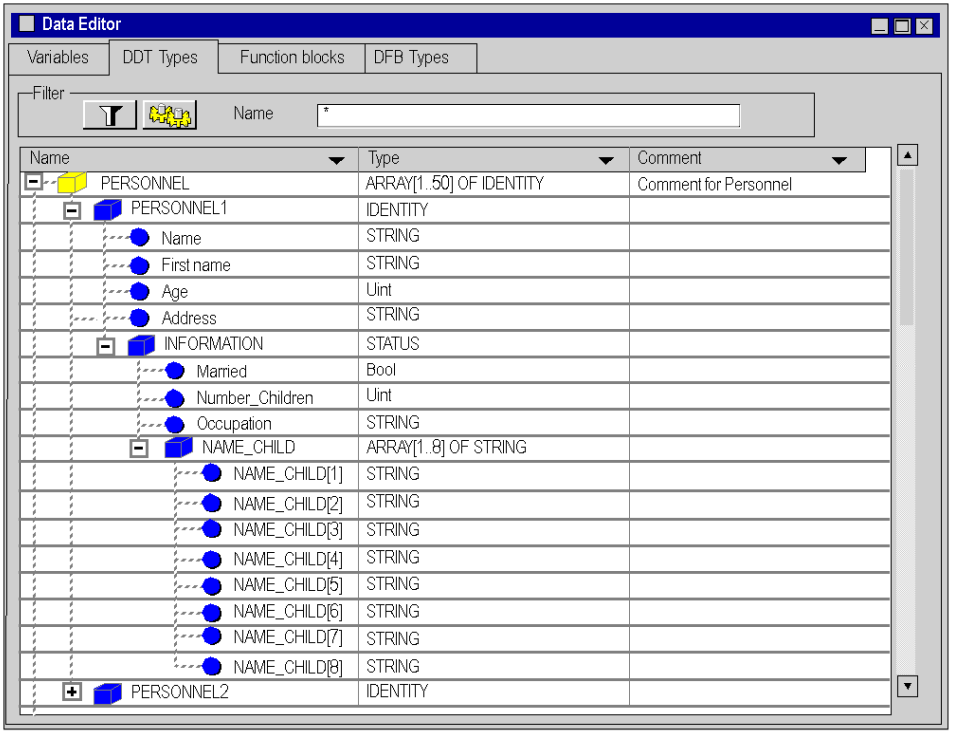


## Combination of the different nestings.

The following example shows:

- a **PERSONNEL** array consisting of **IDENTITY** type structures
- **IDENTITY** structures containing **STATUS** type structures
- **STATUS** structures containing arrays of the **NAME_CHILD** type

Example:



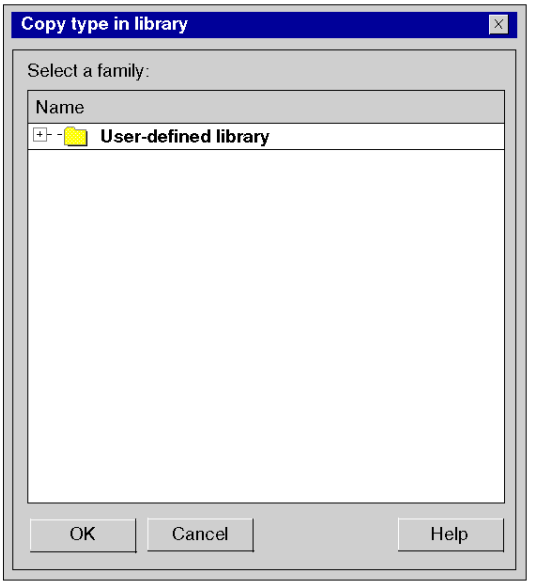
# Save Derived Data Types (DDT)

## Description of how to save

You can copy the data types you created into your library to reuse them in another project.

## Filing a data type in the library

Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the **DDT Types** tab in the Data Editor. |
| 2 | Select the data type that you want to save in the library. |
| 3 | Using the shortcut menu select the **Copy into Library** command.<br>The following dialog box opens:<br>  |
| 4 | Show the desired family to select the directory where you want to archive the data type. |
| 5 | Confirm with **OK**. |

**NOTE:** This procedure can also be started from the project browser in the **Derived Data Types** directory.

### Restoring a data type in the library

See loading an object from the library. *(see page 307)*

# Chapter 11
# Creating data types of user function blocks (DFB)

## Creation and Management of Data Types and Function Blocks (DFB)

### At a Glance

See Derived Function Blocks (DFB) *(see page 1251)*.

This chapter explains the **DFB Types** tab modes of the Data Editor with the following descriptions:

- accessing the DFB Types
- creating a DFB Type
- programming a DFB Type
- managing the elements of a DFB Type
- nestling of the DFB Types
- saving DFB Types

# Chapter 12
# Managing of instances of data belonging to the family of function blocks (EF)

## Object of this chapter

This chapter guides the user through the creation and management of data instances belonging to block function types such as:

- Elementary function blocks (EFB),
- Derived function blocks (DFB).

## What Is in This Chapter?

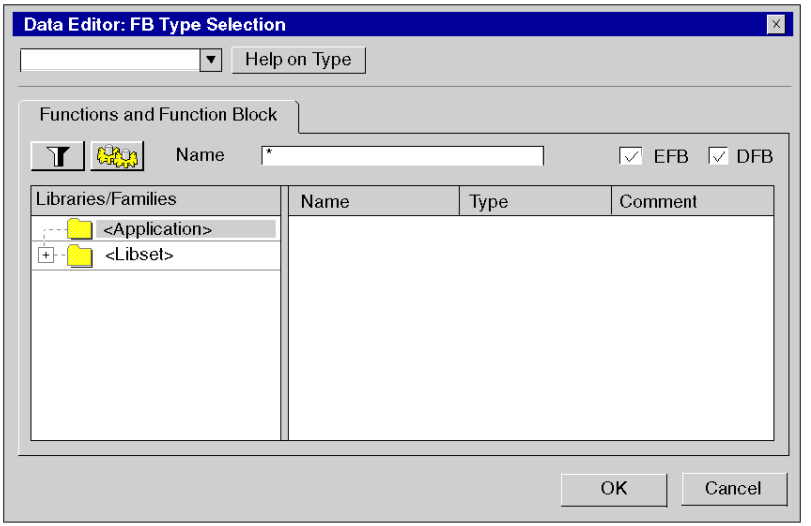
This chapter contains the following topics:

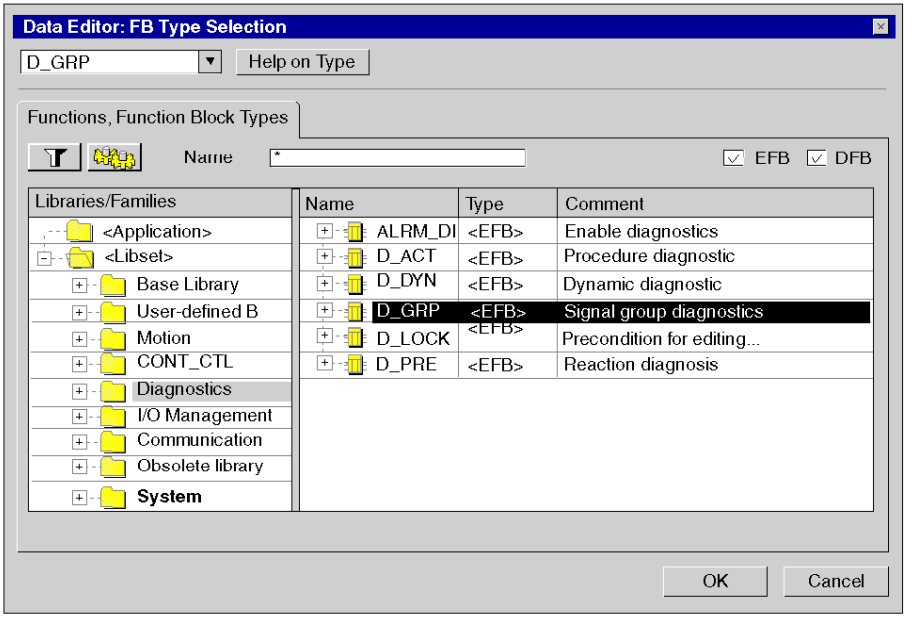

| Topic | Page |
|---|---|
| Creating / Deleting a variable instance of function blocks (EFB/DFB) | 360 |
| Displaying function block variable instances in the editor | 363 |
| Changing the attribute for variable instances in a function block | 366 |

# Creating / Deleting a variable instance of function blocks (EFB/DFB)

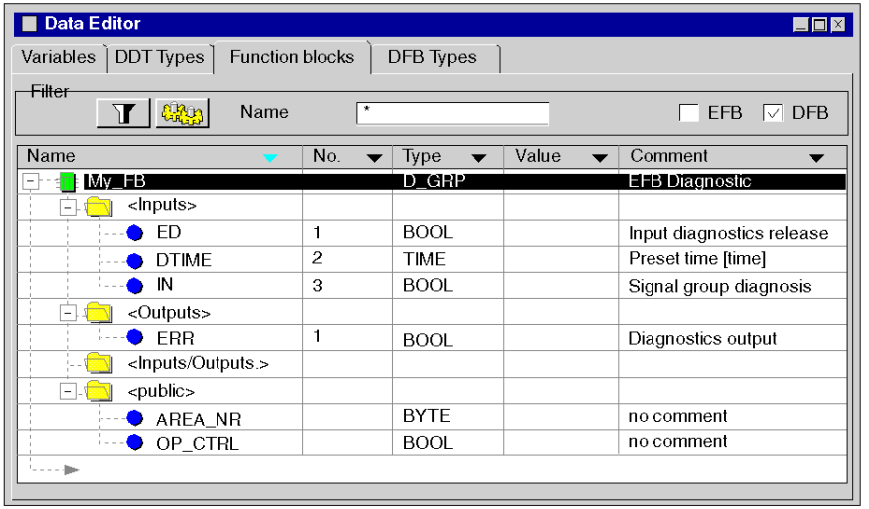
## Creating a function block instance

Carry out the following steps:

| Steps | Actions |
|---|---|
| 1 | In the Data Editor select the **function blocks** tab. |
| 2 | Double-click the empty cell for **Name** (marked with an arrow), enter the name of the instance and confirm by pressing the **Enter**button. |
| 3 | Double-click on the corresponding **Type** field and click on .<br>  <br>The following window appears:<br>  |

| Steps | Actions |
|---|---|
| 4 | In the **libraries/families** area select the following:<br>● the **<Applications>** directory for a type already in the application,<br>● the **<Library>** directory for an already archived type.<br>Example: Select an EFB type archived in the **Diagnostics** library:<br>  |
| 5 | In the **Name** column select the EFB or DFB type and confirm with **OK**. |

Example:




## Deleting a function block instance

Instances that are accessed at least once cannot be deleted in the data editor.

Example from an ST section:

```
FBI_1(IN:=TRUE, OUT=>VarA);
```

The application explained above will block the deletion because `FBI_1` is accessed.

```
VarA:=FBI_1.OUT
```

The application explained above will **not** block the deletion because `FBI_1` is not accessed.

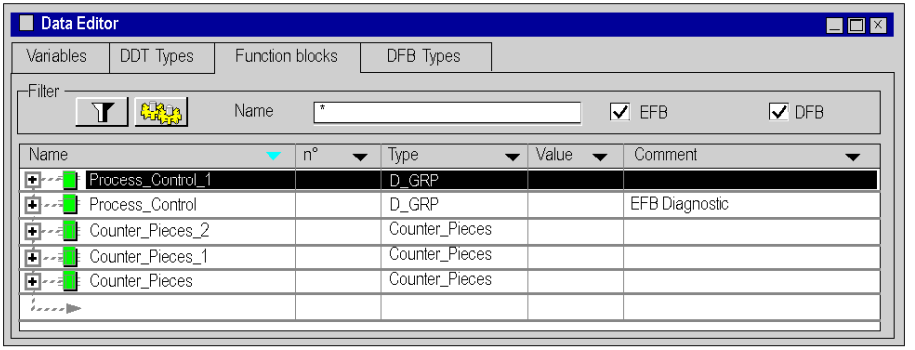
# Displaying function block variable instances in the editor

## Display sequence in a column

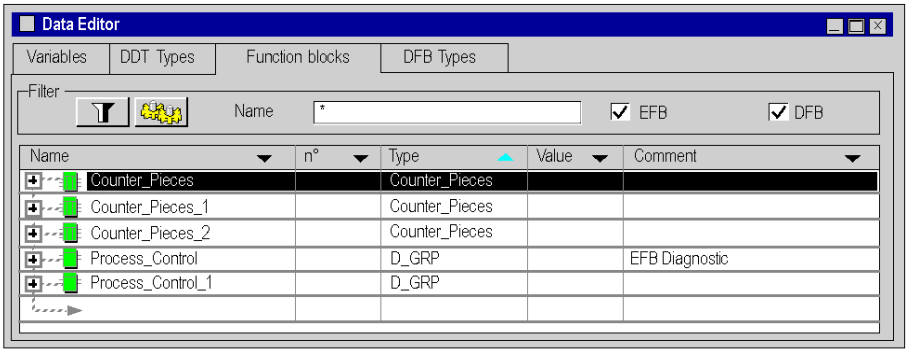
The variable instances can be displayed in **ascending** or **descending alphabetical order**. This sequence will be used in a column chosen by you.

To do this, click on the area that contains the name of the column in question and the arrow will turn blue.

Sorting in descending alphabetical order in the column **Name**




Sorting in ascending alphabetical order in the column **Type**

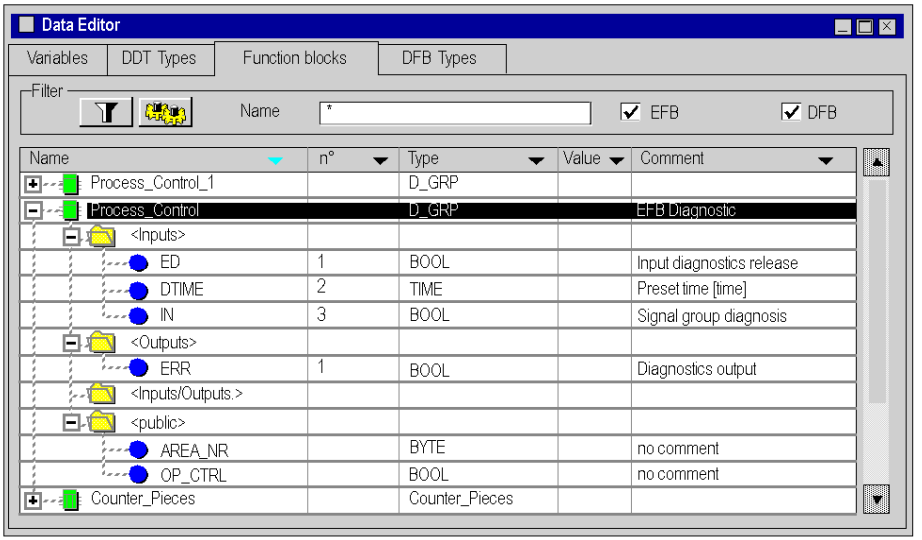


## Displaying elements from one function block instance

To show the elements that belong to an instance of the function block type, carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the Data Editor select the **function blocks** tab. |
| 2 | Click on the **+** beside the instance of the desired function block type and the elements will then be shown on screen. |
| 3 | Carry out the procedure again if you want to display the interface data or the internal data for the function block on the screen. |

Example: Display the instance **Control_Process** of the type EFB

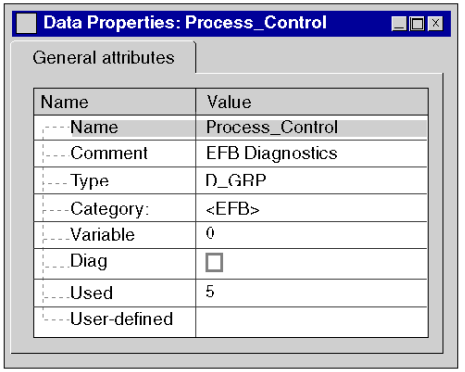


## Displaying the attributes

For an **unopened** window **Data Properties**:

| Step | Action |
|---|---|
| 1 | In the Data Editor select the **function blocks** tab. |
| 2 | Select the variable instance. |
| 3 | From the context menu select the command **Data Properties** and the attributes will be displayed in a window. |

Example:




For a window that is already open **Data Properties**:

After the instance has been chosen, the window **Data properties** is updated automatically with the display for the attribute in question.

## Sorting the variable instances

See *Filtering of Data, page 412*.

# Changing the attribute for variable instances in a function block

## Detailed information

The internal organization of a variable instance of function blocks (input / output parameters, public / private variables, code) cannot be changed from the instances, but must be performed from the type of the instance **DFB Types**) tab.

The changes listed below do not effect the internal organization of the function block, but rather its instantiation.

## Changing the name of an instance

Carry out the following steps:

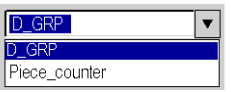
| Step | Action |
|---|---|
| 1 | In the Data Editor select the **function blocks** tab. |
| 2 | Double-click on the instance name, enter the new name, and confirm with **Enter**. |

**NOTE:** the instance name can only be used once.

## Changing the type of an instance

There are three options.

In the Data Editor **Function blocks** tab:

| Double-click on the variable type and ... | then... |
|---|---|
| enter the desired type,<br>  <br><br>depending on the input, an automatic search is carried out. | - |
| use the type field<br>  <br> | select the desired type. |
| click the button  <br> | select the type in the "FB type selection" *(see page 408)* window. |

### Reactions to type changes of an instance.

If the type of an FB instance is changed in the Data Editor, the access will be adjusted to the instance.

The access is adjusted as follows:

- The connections are mapped with pin positions.
  **Note:** Compatibility of data types will not be checked, which means that errors may occur during the analysis following the adjustment.
- Excess pins of the new type remain unused.
- The connections of pins that no longer exist in the new type are lost.

**IL and ST sections:**

If necessary, formal parameters in IL and ST sections will be renamed according to the new block during formal access. The number of actual parameters will be adjusted during informal access.

**NOTE:** Instance access in IL or ST sections will only be adjusted if required by the type change. However, if an adjustment is made, the old access will be removed and replaced with a generated access. This means that all comments and formatting contained in the old access will be lost.

**FBD and LD sections**

The new type will be displayed in the FBD and LD sections following the adjustment.

# Chapter 13
## Managing variable instances belonging to the EDT/DDT/IODDT/Device DDT families

### Object of this chapter

This chapter guides the user through the creation and management of variable instances belonging to the following families:

- Elementary data types (EDT)
- Derived Data Types (DDT)
- Derived Data Types that reference IODDT inputs/outputs (input / output data types)
- Device Derived Data Types (Device DDT)

### What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Creation of EDT type variable instances | 370 |
| Creation of variable DDT type instances | 375 |
| Creating a type IODDT variable instance | 382 |
| Creating a Type Device DDT Variable instance | 385 |
| Managing an IODDT, DDT or an Array type instance | 387 |
| Creation of multiple variable instances | 393 |
| Displaying the variable instances in the editor | 395 |
| Changing attributes of variable instances | 399 |
| Creation of a Reference Type Instance | 402 |

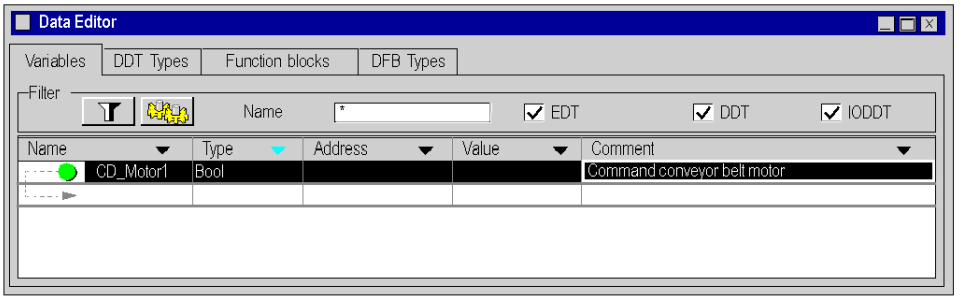
# Creation of EDT type variable instances

## Instance of an Unlocated Variable

Creating the instance of an **unlocated** variable:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click the empty cell **Name** (marked with an arrow) and enter the name of the instance. |
| 3 | Double-click the **Type** cell and enter the type of the instance. |
| 4 | Enter a comment if necessary. |

Example:

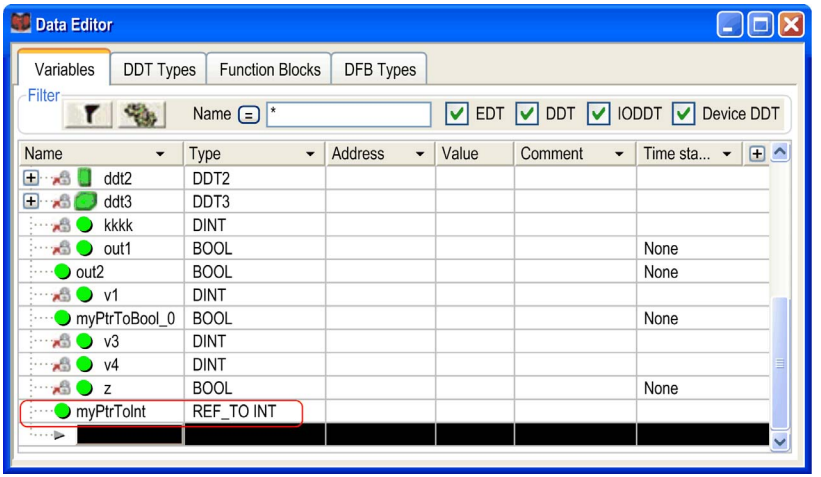


## Instance of a Referenced Variable

Creating an instance of a **Referenced** variable:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click the empty cell **Name** (marked with an arrow) and enter the name of the instance. |
| 3 | Double-click the **Type** cell and enter **REF_TO** and the type referenced for this instance. |
| 4 | Enter the initial value *(see page 372)* of the variable instance. |

Example:

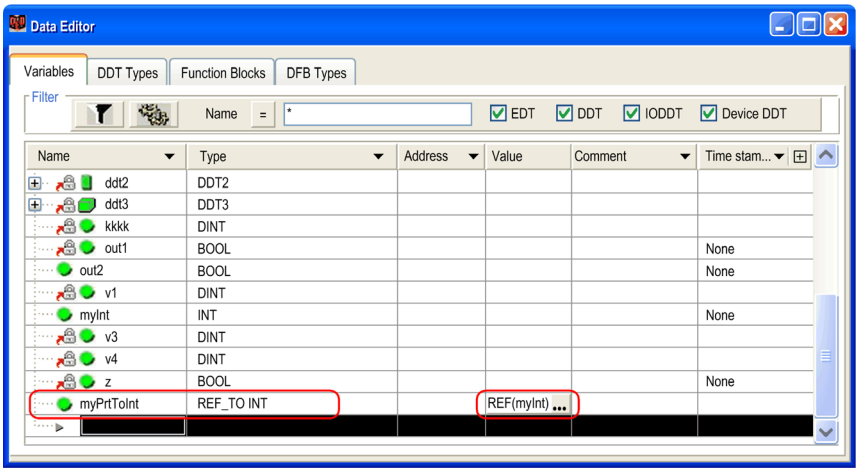


## Initial Value of a Reference Type Variable

To set an initial value to a variable of reference type

| Step | Action |
|---|---|
| 1 | In the **Variables** tab of the **Data Editor**, select the reference type variable to be initialized. |
| 2 | Double-click **Value**. |
| 3 | Enter `REF(VarName)`, where `VarName` is the variable referenced by the Instance of the Referenced Variable. |

Example:



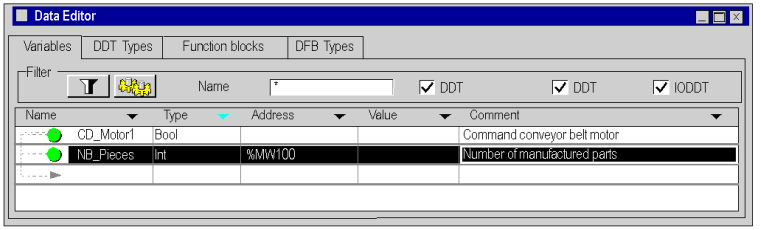
## Instance of a Located Variable

Creating an instance of a **located** variable:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the empty cell **Name** (marked with an arrow) and enter the name of the instance. |
| 3 | Double-click on the **Type** cell and enter the type of the instance. |
| 4 | Double-click on the **Address** cell and enter the storage address for the controller. |
| 5 | Enter a comment if necessary. |

**NOTE:** Several different instances can be assigned to one address (multiple assignment).

Example:

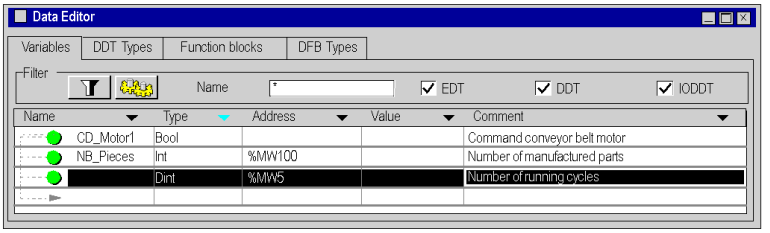


## Instance with Direct Addressing

Creating an instance with direct addressing

| Step | Action |
|---|---|
| 1 | Double-click on the empty **Address** cell and enter the storage address for the controller. |
| 2 | Enter a comment if necessary. |

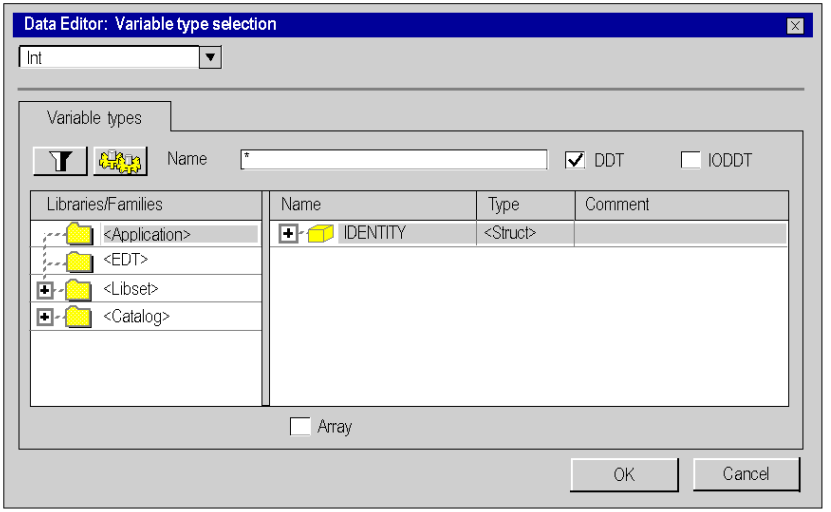
Example:



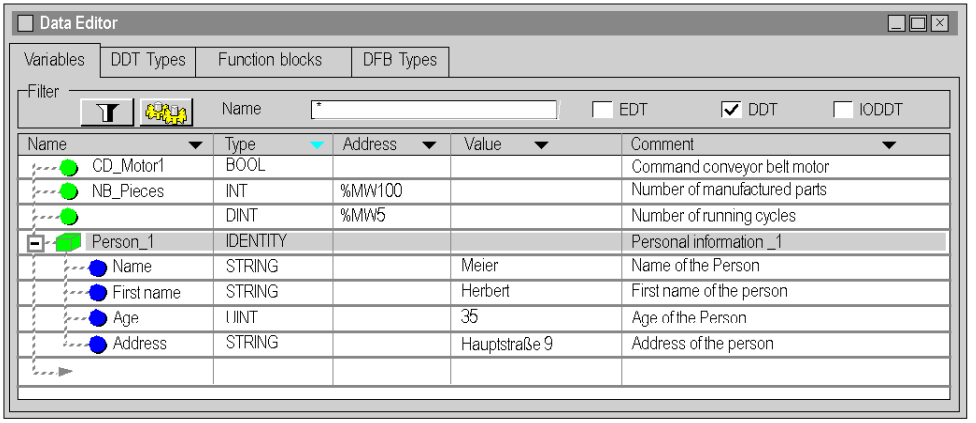
# Creation of variable DDT type instances

## Structure type instance

Creating a Structure type instance:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the empty cell **Name** (marked with an arrow) and enter the name of the instance. Enter a comment (optional). |
| 3 | Double-click on the corresponding **Type** field and click on.<br>  <br>The following window appears.<br>  |
| 4 | In the **libraries/families** area select the following:<br>● the **<Application>** directory for an existing type in the application.<br>● the **<Libset>** directory for an already archived type. |
| 5 | Enable the **DDT** checkbox, which will display the derived data types. |
| 6 | Select the desired structure type in the **Name** column and confirm with **OK**. |
| 7 | Fill out the **Value** column to initialize the elements of the structure. To carry out this task, click on **+** to show the structure. |

Example:




**NOTE:** The comments will be transferred to the selected structure type, but they may be changed for the applicable instance.

## Instance of a non-typical array.

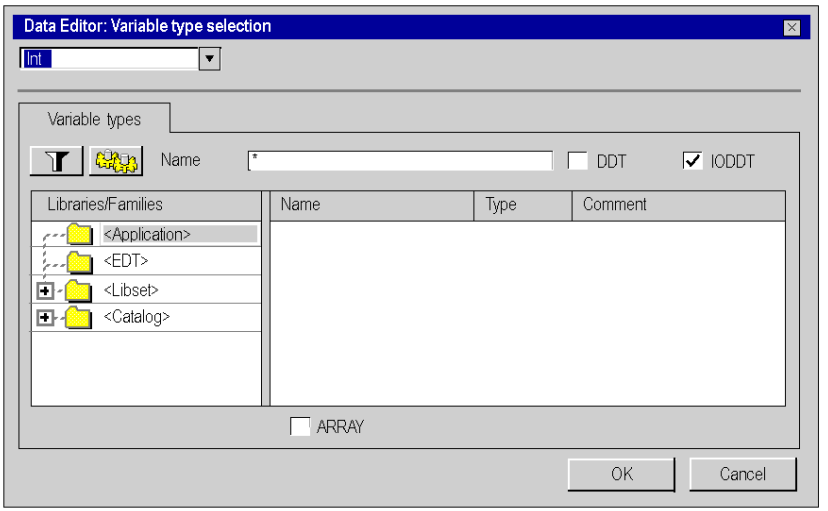
What is a **non-typical** array?

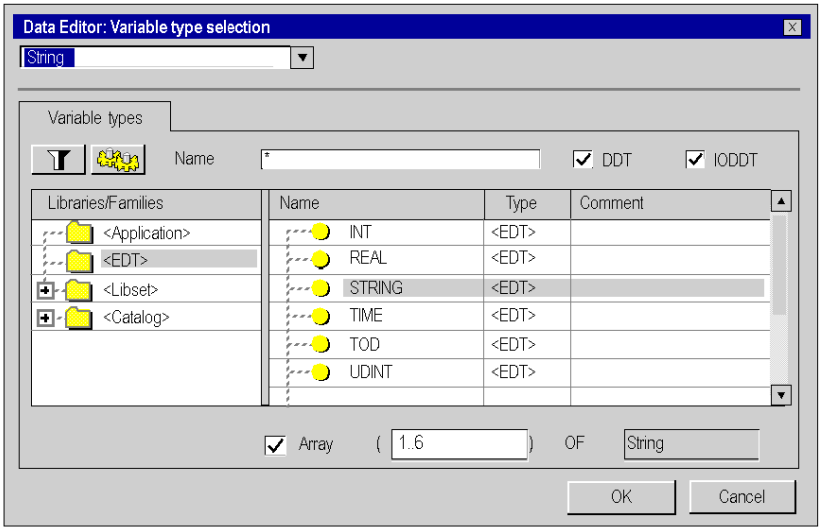
```
Defining an array type
Tab: ARRAY[1..10] of BOOL        ; Array of Type Tab

Array instantiation
My_Tab1 : Tab                     ;typical Array
My_Tab2 : ARRAY[1..10]OF BOOL     ;non-typical Array
```

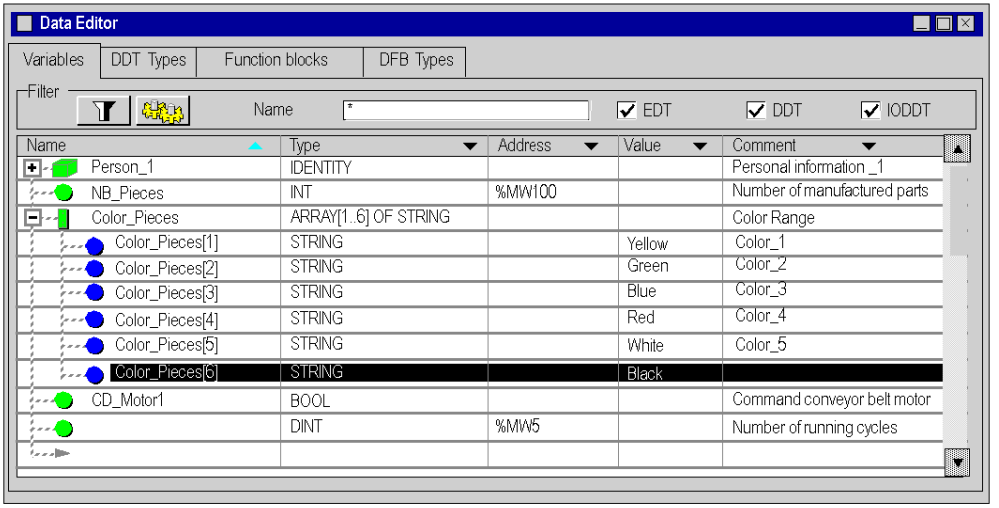
**NOTE:** My_Tab1 and My_Tab2 are two identical instances. The difference is that the type of the My_Tab1 instance was previously defined (the name of the Type is Tab) while the type of the My_Tab2 instance was defined during instancing.

Creating an instance for a **non-typical** array:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the empty cell **Name** (marked with an arrow) and enter the name of the instance. Enter a comment (optional). |
| 3 | Double-click on the corresponding **Type** field and click on.<br>  <br>The following window appears.<br>  |

| Step | Action |
|---|---|
| 4 | Enable the **Array** check box and<br>● fill in the field with the size of the array (example 1..6),<br>● In the **libraries/families** area select the types that should be displayed:<br>❍ the **<EDT>** directory for an elementary type,<br>❍ the **<Application>** directory for an existing derived type in the application,<br>❍ the **<Libset>** directory for a derived type archived in a library.<br>● Select the type for the elements of the array in the **Name** column (Example: `STRING` for an EDT).<br>Example:<br>  <br><br>Confirm with **OK**. |
| 5 | Fill out the **Value** column to initialize the elements of the array as needed. To carry out this task, click on **+** to show the elements. |

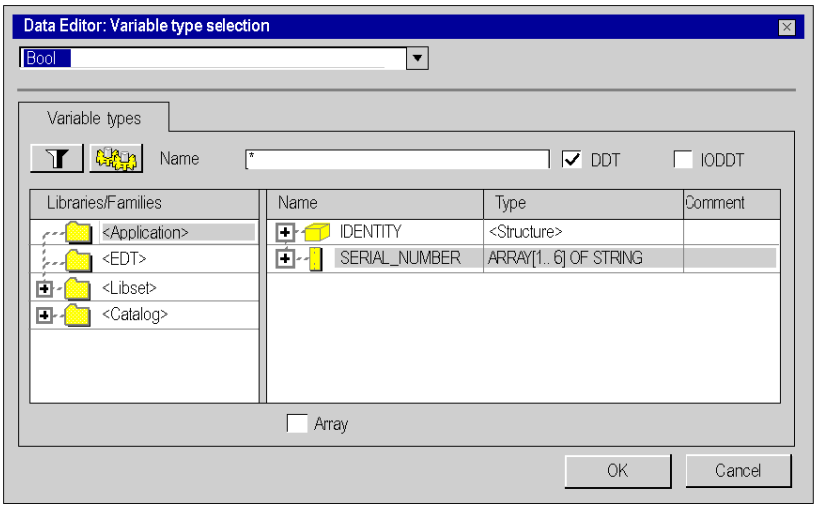
Example:



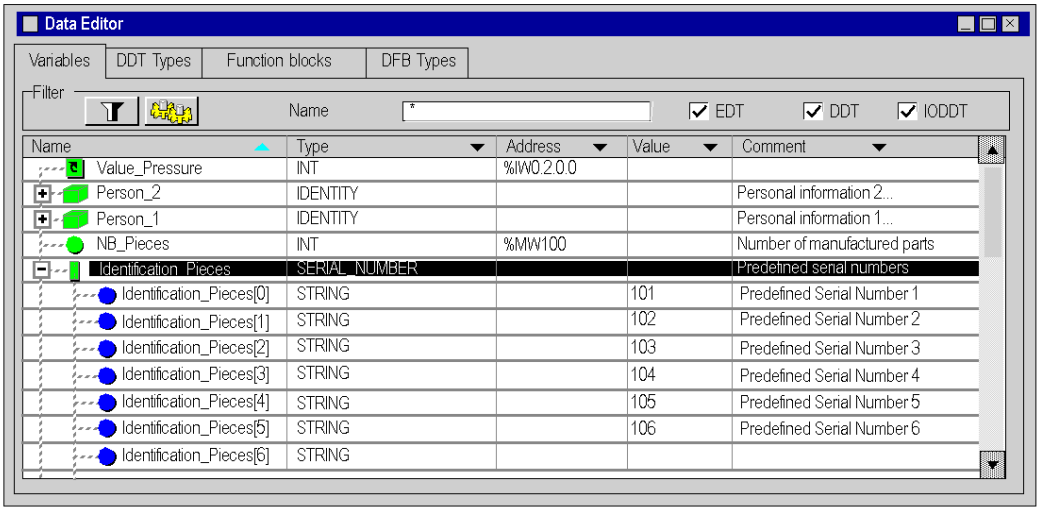

**NOTE:** Enter comments for the sub-elements of this instance of a **non-typical** array.

## Type Array instance

Creating an Array type instance:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the empty cell **Name** (marked with an arrow) and enter the name of the instance. Enter a comment (optional). |
| 3 | Double-click on the corresponding **Type** field and click on.<br>  <br>The following window appears.<br>  |
| 4 | In the **libraries/families** area select the following:<br>● the **<Applications>** directory for a type already in the application,<br>● the **<Libset>** directory for a derived type archived in a library. |
| 5 | Enable the **DDT** checkbox, which will display the derived data types. |
| 6 | Select the desired array type in the **Name** column and confirm with **OK**. |
| 7 | Fill out the **Value** column to initialize the elements of the array as needed. To carry out this task, click on **+** to show the elements. |

Example:




| Name | Type | Address | Value | Comment |
|---|---|---|---|---|
| Value_Pressure | INT | %IW0.2.0.0 | | |
| Person_2 | IDENTITY | | | Personal information 2... |
| Person_1 | IDENTITY | | | Personal information 1... |
| NB_Pieces | INT | %MW100 | | Number of manufactured parts |
| Identification_Pieces | SERIAL_NUMBER | | | Predefined serial numbers |
| Identification_Pieces[0] | STRING | | 101 | Predefined Serial Number 1 |
| Identification_Pieces[1] | STRING | | 102 | Predefined Serial Number 2 |
| Identification_Pieces[2] | STRING | | 103 | Predefined Serial Number 3 |
| Identification_Pieces[3] | STRING | | 104 | Predefined Serial Number 4 |
| Identification_Pieces[4] | STRING | | 105 | Predefined Serial Number 5 |
| Identification_Pieces[5] | STRING | | 106 | Predefined Serial Number 6 |
| Identification_Pieces[6] | STRING | | | |

**NOTE:** Comments cannot be entered for the sub elements of this instance of an array.

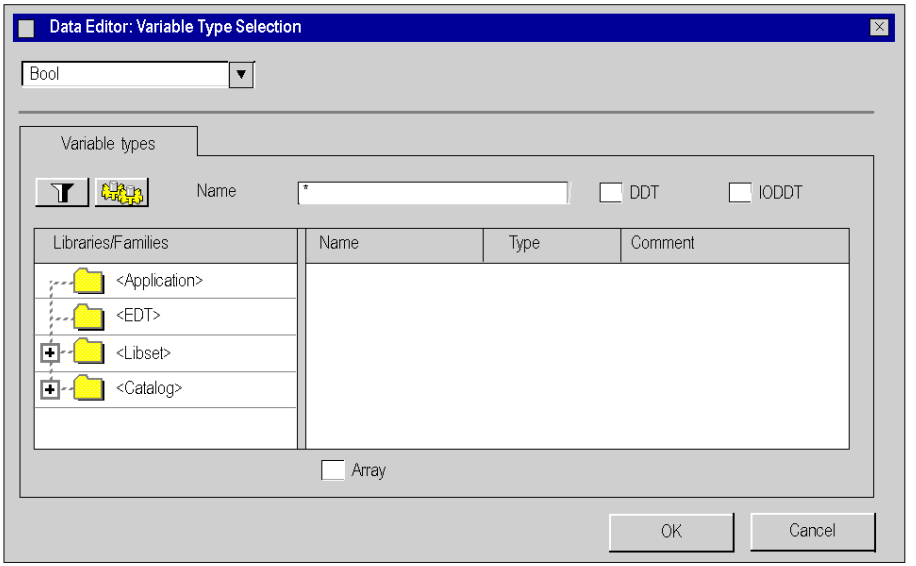
**NOTE:** If you resize an array variable type, you will lose the comments and the initial values of the elements of the array.

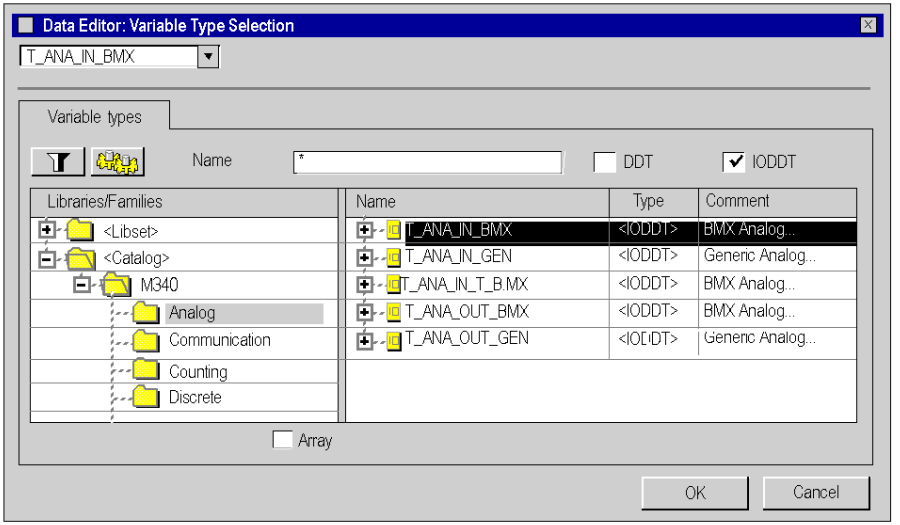
# Creating a type IODDT variable instance

## Type IODDT instance

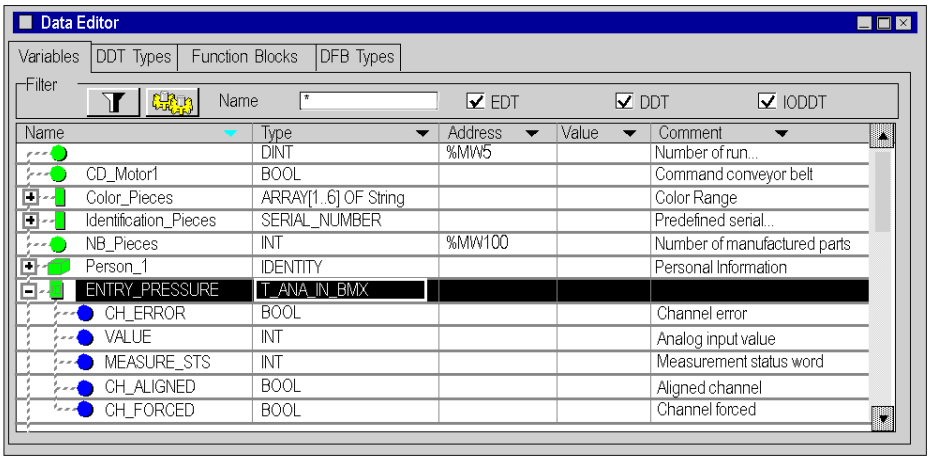
The data types (structures) have been **predefined by the manufacturer** and contain I/O language objects belonging to the channel of an application specific module (or the module itself).

Perform the following steps to create a type IODDT instance:

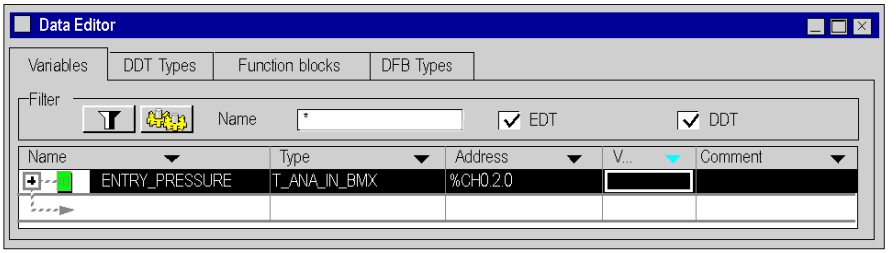
| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the empty cell **Name** (marked with an arrow) and enter the name of the instance. |
| 3 | Double-click on the corresponding **Type** field and click on it .<br>  <br>The following window appears.<br>  |

| Step | Action |
|---|---|
| 4 | In the **Library/Family** area, display the **<Catalog>** index.<br>The following window appears.<br>  |
| 5 | Select the Type of Controller, Function and the corresponding IODDT Type.<br>Confirm with **OK**. |

Example: Instance **ENTRY_PRESSURE** of Type **T_ANA_IN_BMX**:



## Linking to the application specific module

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Enable the **IODDT** checkbox, which will display instances of Type IODDT. |
| 3 | Select the IODDT instance based on which the link will be created. |
| 4 | Enter the address of the module or its future storage location in the **Address** column (it is not necessary to configure the module during this phase).<br>Example:<br>  |

## Rules

Follow the rules below:

- An IODDT data type cannot be nestled in a DDT data type.
- An IODDT data type cannot be nestled in another IODDT data type.
- A public or private variable of a DFB cannot be an IODDT type.
- An input / output parameter of a DFB may be of Type IODDT. However, you can only use those IODDT types that are listed in the **Variable Type Selection** menu.
- An IODDT instance cannot be accessed from a DFB section.

# Creating a Type Device DDT Variable instance

## Type Device DDT Instance

When adding an I/O module to the remote rack the Unity Pro software will automatically instantiated the associated Device DDT structure.

Each I/O modules are associated with one implicit device DDT instances.

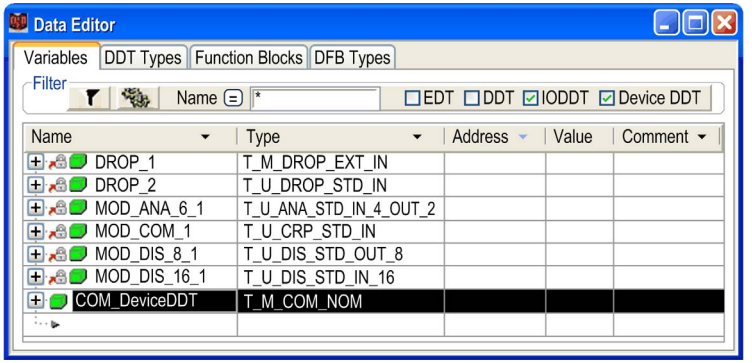
Implicit Device DDT instances are created by default on device insertion and refreshed automatically by the PLC. They contain the modules status, modules and channels health bits, values of the modules inputs, values of the modules outputs, etc.

**NOTE:** Optional explicit DDT types are proposed in the data editor to support status or command data used by explicit exchanges on a channel of an IO Modules in a remote I/O drop. Explicit DDT instances are created manually by the user in the data editor and used as input or output parameter with the function block managing the explicit exchange *(see Unity Pro, I/O Management, Block Library)*.

Perform the following steps to create a type Device DDT instance:

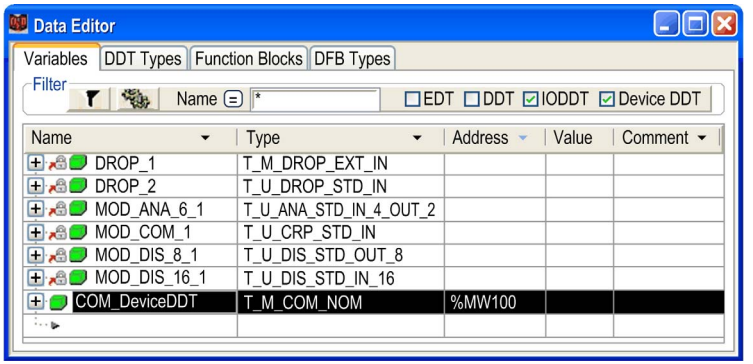
| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the empty cell **Name** (marked with an arrow) and enter the name of the instance. |
| 3 | Double-click on the corresponding **Type** field and click on it.<br>  <br>**Results:** The Variable Type Selection dialog appears. |
| 4 | In the **Library/Family** area, display the **<Libset vx.0>** index.<br>Where **x** indicated the Unity Pro software version. |
| 5 | Select the **I/O Management** and the corresponding Device DDT Type. |
| 6 | Confirm with **OK**. |

Example: Instance **COM_DeviceDDT** of Type **T_M_COM_NOM**:




## Linking to the Application Specific Module

Use this procedure:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Enable the **Device DDT** checkbox, which will display instances of Type Device DDT. |
| 3 | Select the Device DDT instance based on which the link will be created. |
| 4 | Enter the address of the module or its future storage location in the **Address** column (it is not necessary to configure the module during this phase).<br>Example:<br>  |

# Managing an IODDT, DDT or an Array type instance

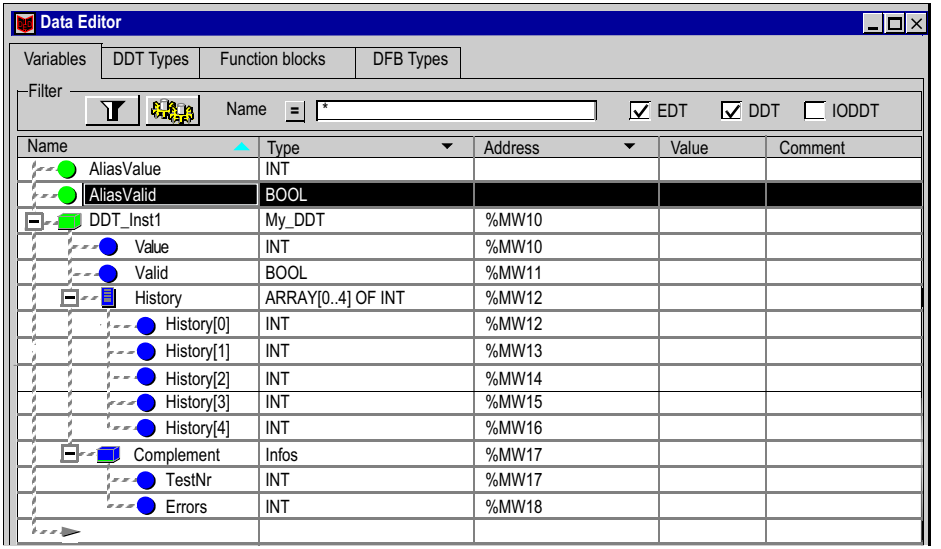
## Adding an alias

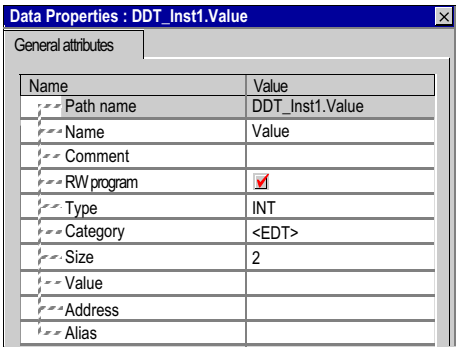
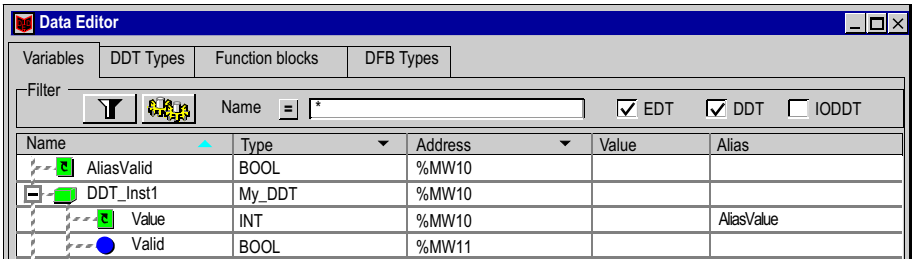
An alias name can be given to all elements of an IODDT, DDT or an Array type instance of application.

The aliases can be created with the following rules:

- The alias descriptions are only allowed for elements of type IODDT, DDT or an Array.
- The **type** and **address** for an alias cannot be changed as these attributes are defined by the IODDT, DDT or an Array
- **Name** and **Comment** can be changed.
- The alias displays the initial value of a DDT or Array element.

This procedure below is used for customizing elements used to make up a DDT instance, however the procedure is the same for an IODDT or Array instance.

| Step | Action |
|---|---|
| 1 | Create one or more variable instances you want to customize that are identical with the elements of the IODDT / DDT / Array.<br>For example **AliasValue** and **AliasValid**:<br>  |
| 2 | Select an element of the **DDT_Inst1** instance (for example **Value**) that you want to assign to the previously created **AliasValue** instance.<br>Alternatively, the user can also assign a new symbol as alias. In this case the alias variable is automatically be created with the type of the current element. |

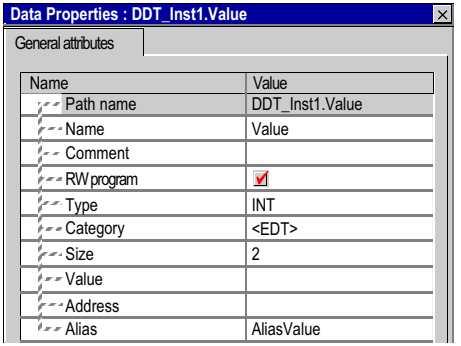
| Step | Action |
|---|---|
| 3 | Using the shortcut menu select the **Properties** command.<br>The **Data properties** field opens:<br>  |
| 4 | In the **Alias** area enter the element **AliasValue** and then do the same for all elements that you want to assign.<br>Along with the alias names and the elements of the DDT a symbol is shown.<br>  |

**NOTE:** The user can directly enter the AliasValue in the Alias column.

## The "Data Properties" window

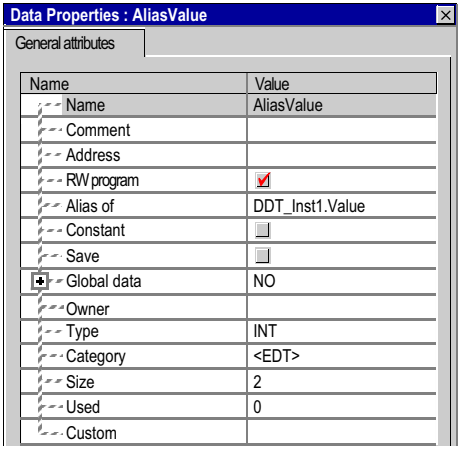
The **Data properties** window can be opened from the **IODDT structure**, **DDT structure** or an **Array** element. In such a case **Alias** is the structure attribute; it contains the names of the variables.

Example:

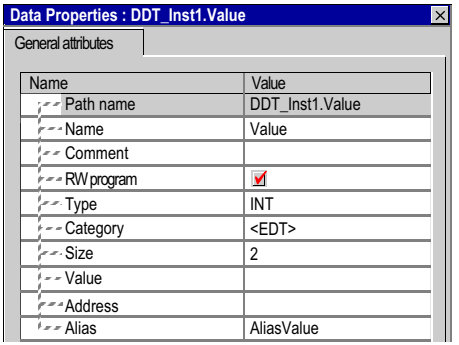
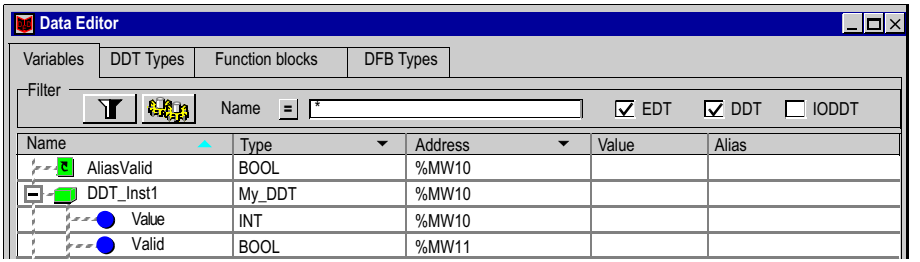



The **Data properties** window can also be opened from the **Alias instance**. In such a case **Alias of** is an instance attribute; it contains the names of the elements of the IODDT structure, DDT structure or an Array element.

Example:



## Disconnecting an alias

| Step | Action |
|---|---|
| 1 | Select the element from the appropriate IODDT structure, DDT structure or Array element. |
| 2 | From the shortcut menu select the command **Properties**, and the following dialog box will appear.<br>Dialog box:<br>  |
| 3 | Double-click on the target alias, delete the element in edit mode (AliasValue) and confirm.<br>The symbols shown on the element disappear:<br>  |

**NOTE:** Alternatively, the user can directly delete the name in the column Alias or AliasOf.

**NOTE:** The alias instance returns to being a normal EDT instance and the attributes for the element to which the instance is assigned are retained. The element for the IODDT structure, DDT structure or an Array element contains the original manufacturer-defined name again.

## Alias on sub fields of existing Alias

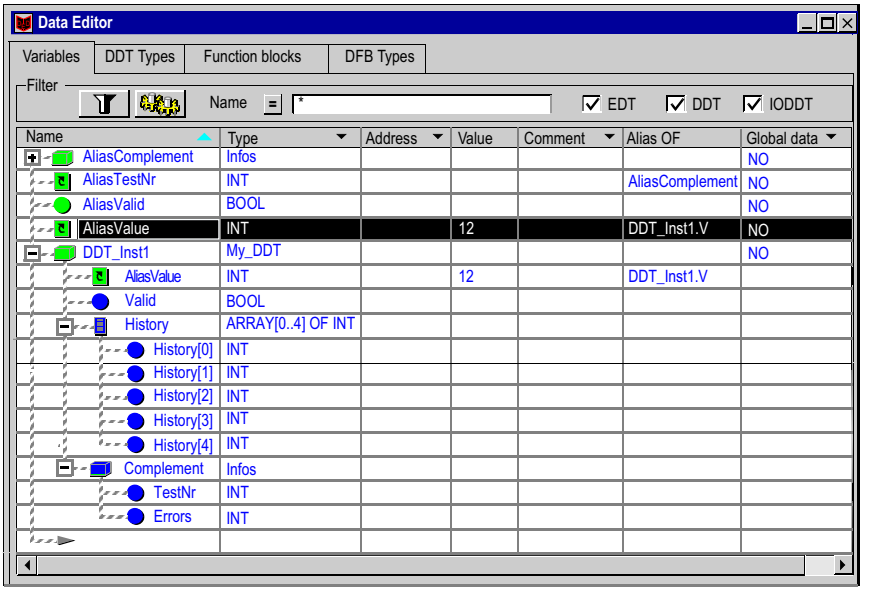
The user can set the alias on a sub field of an IODDT structure, DDT structure or an Array element that is already an alias.

Example: **AliasComplement** is alias of **DDT_Inst1.Complement** and **AliasTestNr** is alias of **AliasComplement.TestNr**.

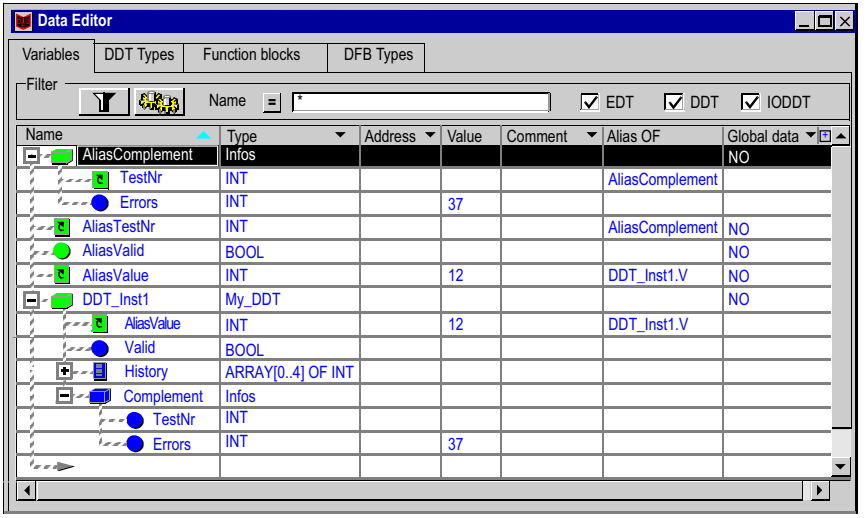
## Management of initial value for an alias

The initial value of an alias is the initial value on the sub field of DDT or Array. This sub field path is given by attribute **Alias OF**. So when you get or set the initial value on an Alias in fact you get it or set it on the sub field path given by attribute **Alias OF**.

Example: Initial value of alias "AliasValue" is the initial value given on path "DDT_Inst1.Value". (12 in the following example screen)



Additionaly if we get or set initial value of “ DDT_Inst1.Complement.Errors” you will get or set initial value on “ AliasComplement.Errors”, because “ AliasComplement” is an alias on “ DDT_Inst1.Complement” (see following figure).



# Creation of multiple variable instances

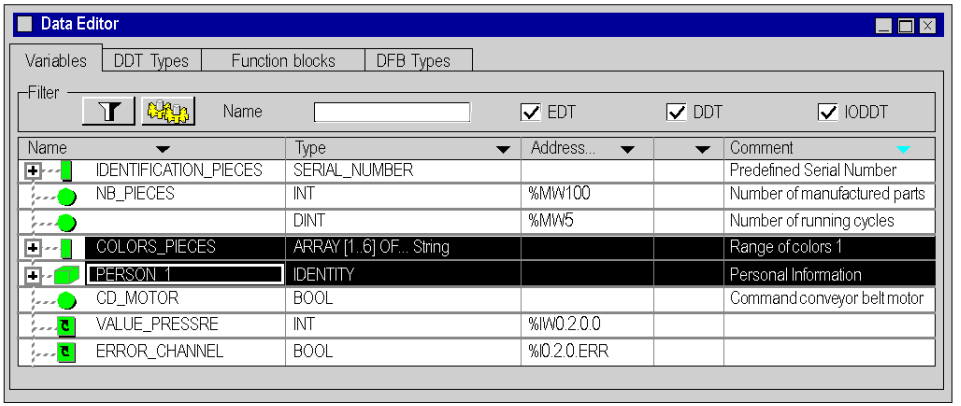
## Procedure

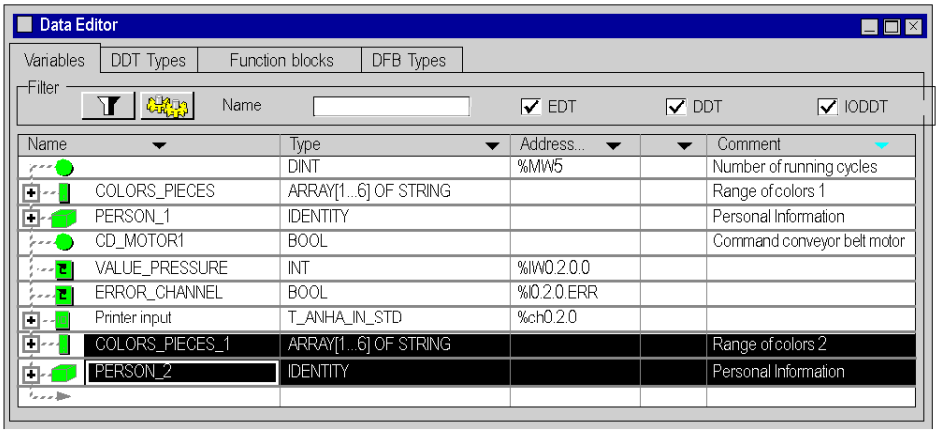
There are two possibilities for this:

- using the **Copy/Paste** function,
- using the **Import** function.

## Copying / pasting of instances.

The following procedure shows the method to create and rename an instance, following defined rules, using the **Copy/Paste** function:

| Step | Action |
|---|---|
| 1 | Select the **Variables**tab in the data editor. |
| 2 | Select the desired instances (rows).<br>  |
| 3 | Using the context menu select the **Copy** command. |
| 4 | Select an open position (row) from which the instances will be copied. |

| Step | Action |
|---|---|
| 5 | From the context menu select the command **Insert** and the new instances will be created and renamed.<br>Example:<br>  |



**NOTE:** Renaming Rules:

- From a name of the instance **<Name>_<Number>** or **<Name>** results after renaming **<Name>_<Number+1>** or **<Name_0>**.

## Importing instances

See "Importing Variables" *(see page 1744)*.

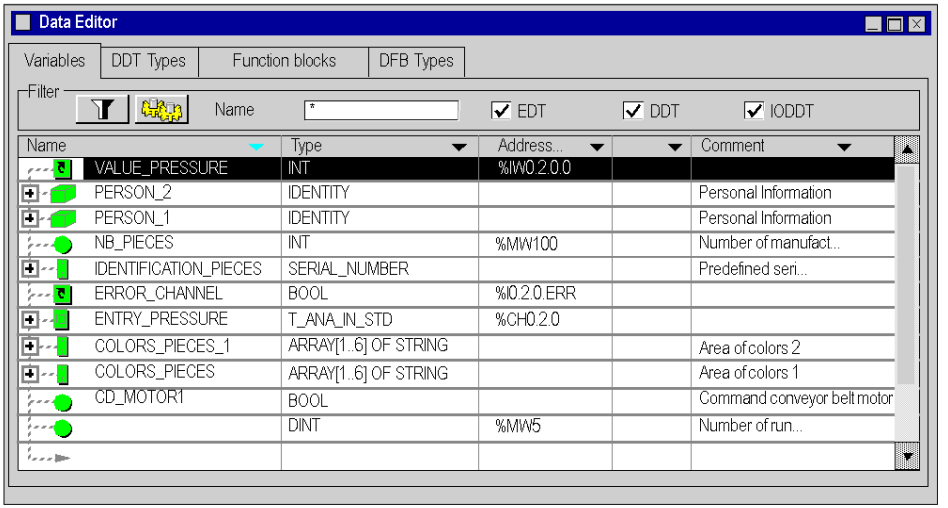
# Displaying the variable instances in the editor

## Display sequence in a column

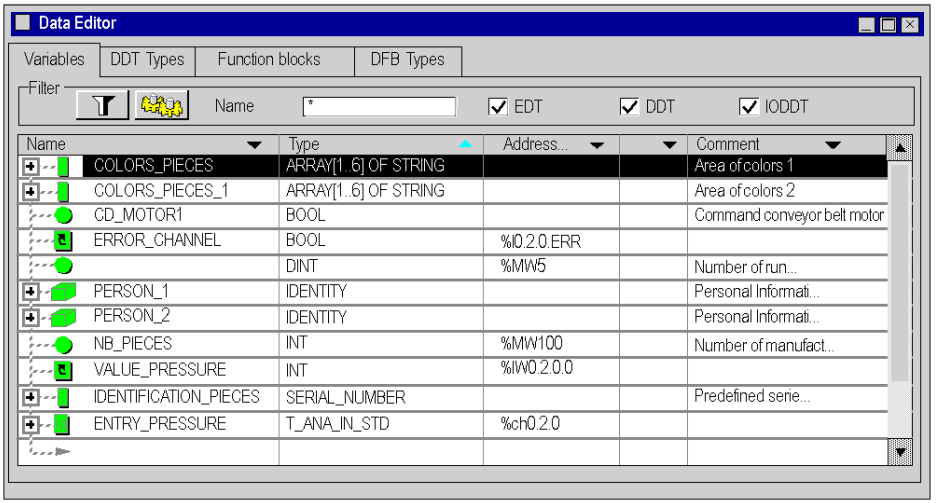
The variable instances can be displayed in **ascending** or **descending alphabetical order**. This sequence will be used in a column chosen by you.

For this, click on the area that contains the name of the column in question which determines the sorting order. After that the corresponding arrow will turn blue.

Sorting in descending alphabetical order in the column **Name**



Sorting in ascending alphabetical order in the column **Type**

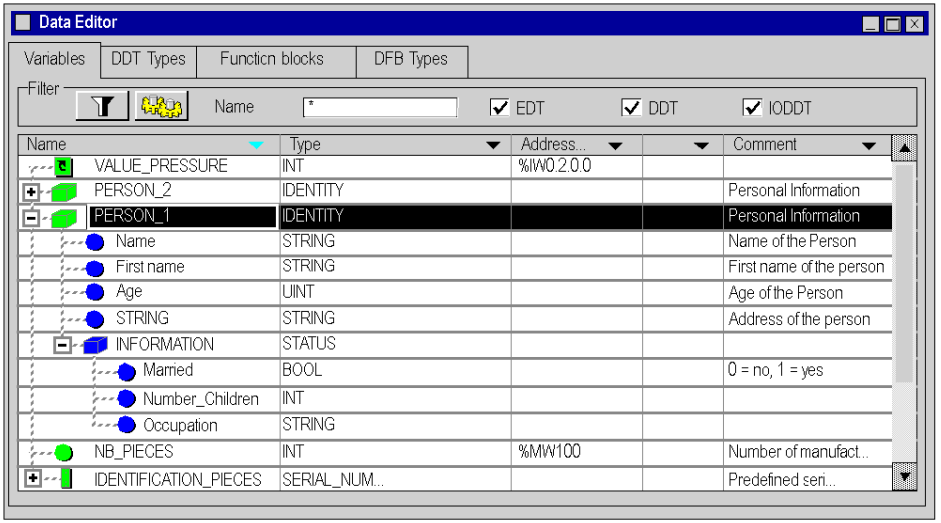



## Display the elements of an instance of the "derived" type

To show the elements that belong to an instance of the "derived" type (structure or array), carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Enable the **DDT** check box. |
| 2 | Click on the **+** beside the instance of the desired structure or array type and this will then be shown on screen. |
| 3 | Run the operation again if you want to display an interlaced element. |

Example: Display the instance `Person_1` of the structure type `IDENTITY`.

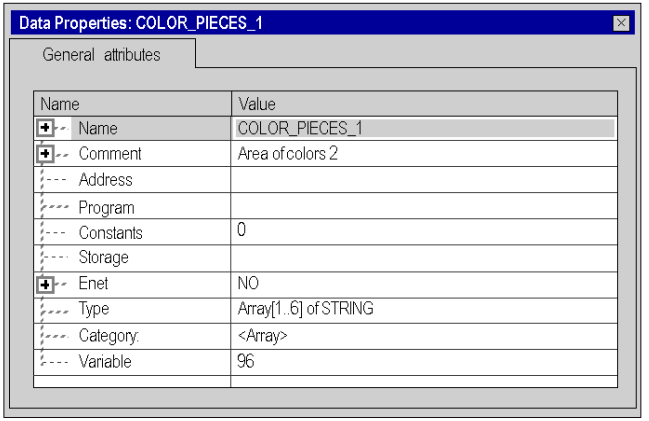



## Displaying the instance attributes

For an**unopened** field **Data Properties**:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Select the desired instance. |
| 3 | Select the **Properties** command from the shortcut menu and the attributes will be displayed in a window. |

Example:




For a window that is already open **Data Properties**:

After the instance has been chosen, the window **Data properties** is updated automatically with the display for the attribute in question.

## Sorting the variable instances

See *Filtering of Data, page 412*

# Changing attributes of variable instances

## Changing the name of an instance

Carry out the following steps:

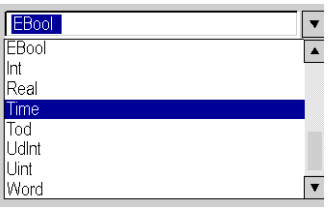
| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the instance name and enter the new name. |

**NOTE:** The instance name can only be used once.

## Changing the type of an instance

There are three options.

In the Data Editor select the **Variables** tab:

| Double-click on the variable type and ... | then... |
|---|---|
| enter the desired type   | |
| use a Type from the list   | select the desired type |
| click the button   | select the appropriate type in the "Variables type selection" *(see page 408)* window |

## Changing the address of an instance

Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Double-click on the instance address and enter the new address. |

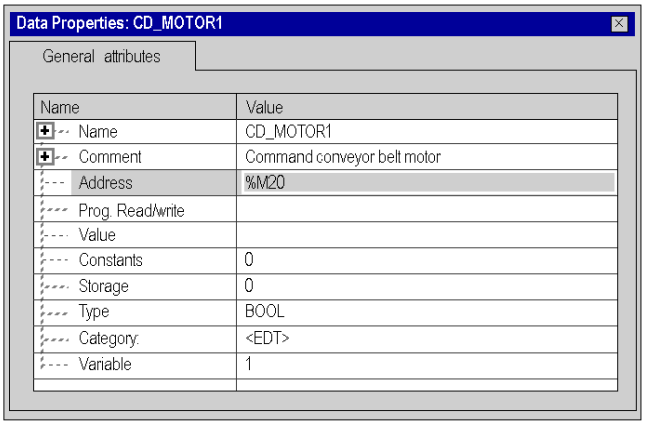
**NOTE:** Several different instances can be assigned to one address (multiple assignment).

## Changing an attribute in the "Data Properties" field

Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Select the desired instance. |
| 3 | Use the shortcut menu to select the **Properties** command. |
| 4 | In the **Data properties** field select the desired attribute. |
| 5 | Change the attribute. |

Assigning an address for the CD_MOTOR1 instance (the variable is located):




## Changing the attribute in several instances

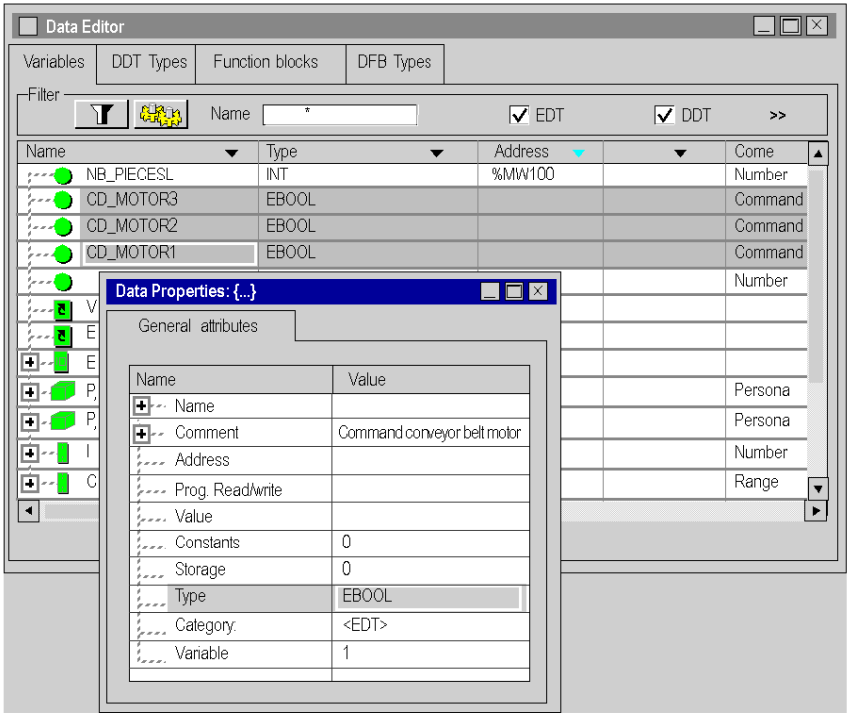
Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the data editor. |
| 2 | Select the desired instances. |
| 3 | Use the shortcut menu to select the **Properties** command. |
| 4 | Select the desired attribute in the **Data Properties** window. |
| 5 | Change the attribute. |

**NOTE:** For this functionality the following restrictions apply:

- names and addresses cannot be changed,
- the contents of the attribute to be changed must be the same for all selected instances,
- the change must be tolerated by all the instances selected,
- for sub-element instances with different superior elements, multiple selections are not possible.

Example: As can be seen in the following graphic, the three instances have the same type and can be changed from Bool to Ebool.




## Rule for instances of the type Structure

The **internal** organization of an instance of type structure **(type, that belongs to the DDT family)** cannot be changed via the instance, but rather via the type of the instance **(DDT type tab)**.

In contrast, the changes listed above **(instance name, type, address)** are applied directly to the instances and do not affect the internal organization, only the instantiation of the type.

# Creation of a Reference Type Instance

## Reference Type Instance

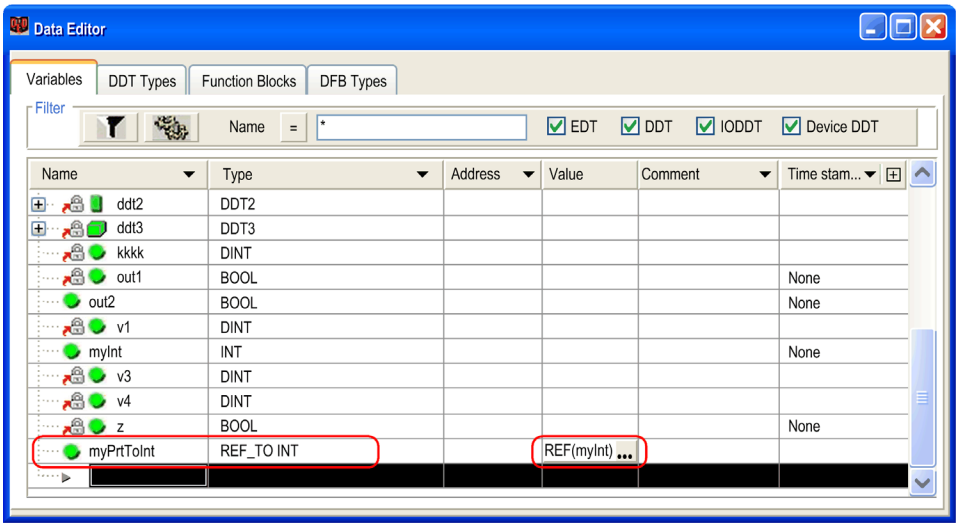
The reference type is used to create references to EDT, DDT and Device DDT variables.

## Creating a Reference to a Variable

Creating a reference type instance:

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab in the **Data Editor**. |
| 2 | Double-click on an empty cell **Name** (marked with an arrow) and enter the name of the instance. Enter a comment (optional). |
| 3 | Double-click on the corresponding **Type** field and click.<br>  <br>The **Data Editor: Variable Type Selection** dialog opens. |
| 4 | Click the **REF_TO** checkbox. |
| 5 | Enable the **EDT**, **DDT** or **Device DDT** checkbox. |
| 6 | Select the desired type in the **Name** column and confirm with **OK**. |
| 7 | Fill out the **Value** column to initialize the variable by typing the name of the variable to be referenced. |

Example of the **Data Editor** with a reference to a variable:



# Chapter 14
# General Functions

## Object of this chapter

This chapter guides the user through the use of the general features available in the data editor.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Configuring the Data Editor columns | 404 |
| Description of the Data Editor: Variable type selection | 408 |
| Filtering of Data | 412 |
| Exporting a subset of variables | 416 |
| Analyze DDT and DFB Data Types | 417 |
| Purge of Unused Instances | 418 |
| Local printing of data | 420 |
| Save the context of the Data Editor | 421 |

# Configuring the Data Editor columns

## Introduction

In every Data Editor tab you can configure the column display (a default configuration is suggested). This lets you:

- define the columns you want to display,
- define the sequence for the columns shown.

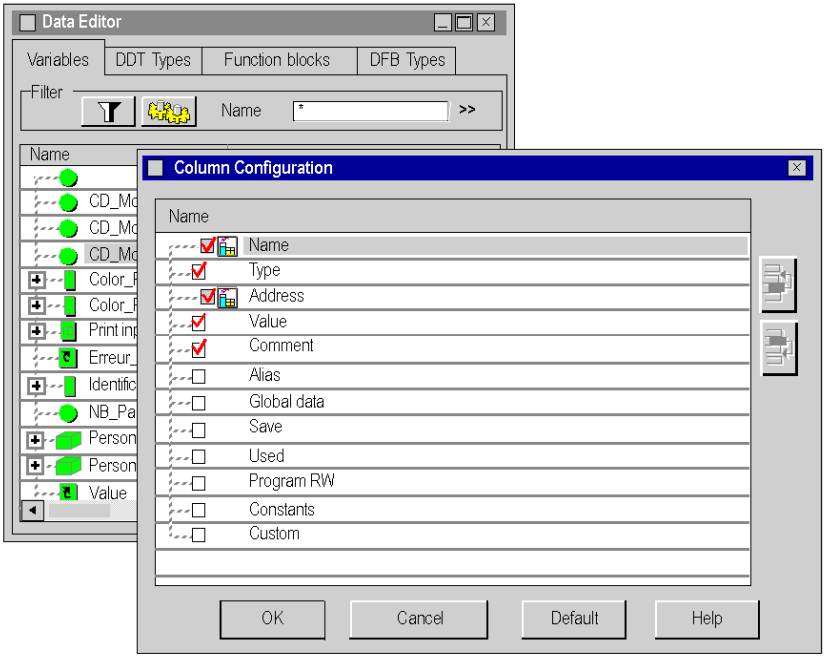
## Columns Available

The following table showns the different columns available in the Data Editor under various tabs.

| Column Name | Description |
|---|---|
| **Name** | name of the instance or data type |
| **Type** | data type |
| **Address** | address in the controller for referencing the variables |
| **984 Address** | LL984 address of the variable (%Mx = 00000x, %Ix = 10000x, %IWx = 30000x, %MWx = 40000x)<br>This column is only available for Quantum PLCs (but not for Quantum Safety PLCs). |
| **Value** | initial value for the variables |
| **Comment** | comment for the variables |
| **Custom** | free text for use of an external tool or for special user requirements |
| **Global data** | Shows if the variable is a variable that was swapped between different stations via the network. |
| Global data<br>**- Group** | number of the group (the variables) in which the variable is transported |
| Global data<br>**- ID** | number for the unique display of a variable in the group (the variables) |
| **Alias** | name selected by the user which makes it possible to change an IODDT, DDT or an Array element to a name defined by the manufacturer or the user |
| **Alias of** | complete path of the variable sub-field on which user sets Alias attribute |
| **Save** | define if the variable accepts the mechanism for updating initial values with current values. |
| **Used** | number of times the variables are used in the program and in other parts of the project (animation tables, user windows) |
| **RW program** | access rights of the variable, read only or read and write by the program. |
| **Constant** | kind of variable, the variable is a constant or not. |

| Column Name | Description |
|---|---|
| **Owner** | only for **Device DDT**, it displays the address of the modules which owns this variable. |
| **Scanned Variables** | attribute that you can use to tag a variable to be a scanned variable by the PLC, through Ethernet for instance. |
| **Exchange On STBY** | Only for M580 Hotstandby systems *(see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures)*, if checked, the variable or the function block instance is automatically exchanged between Primary PLC and Standby PLC. By default this attribute is checked. |
| **Task** | Give the task in which a Variable, EFB instance or DFB instance is used. |
| **Retain** | If checked, the variable, EFB instance or DFB instance values are automatically stored in the backup memory, if not checked the values are lost on power cut. By default this attribute is checked. |
| **Descriptor** | descriptors (1 to 9) of variables<br>**Descriptor 1** is displayed after selecting this column in the **Column Configuration** dialog. **Descriptors 2** to **9** can be displayed using the **+** sign in the **Descriptor** column header. |
| **No.** | order number of a function block |
| **Version** | type version |
| **Bit rank** | gives the bit rank of the variable in its structure type. |
| **HMI** | when **General Project Settings → Data Dictionnary → Only HMI variables** is checked, this reduces memory usage. |
| **Time stamping** | select if a variable is time stamped or not and the kind of time stamping, see the System time stamping variables *(see System Time Stamping, User Guide)* or Applicative time stamping variables *(see Applicative Time Stamping with Unity Pro, User Guide)*. |
| Time stamping<br>**- Source** | gives the source of the time stamping |
| Time stamping<br>**- TS ID** | gives the ID of the time stamped variable |
| **R/W Rights of Referenced value** | access rights of a Reference type instance, used when the reference dereference a read only variable, see the Reference Access rights *(see Unity Pro, Program Languages and Structure, Reference Manual )*. |

## Defining Columns

| Step | Action |
|---|---|
| 1 | Select the desired tab in the Data Editor. |
| 2 | Via the context menu open the **Column Configuration** dialog using **Customize Columns.**<br>The following dialog appears.<br>  |
| 3 | Check the check boxes that correspond to the columns you wish to display. |
| 4 | Confirm the changes using the command **OK.** |

**NOTE:** The **Name** and **Type** columns are required.

Description of the buttons

| Button | Description |
|---|---|
| **OK** | confirms the configuration and exits the dialog |
| **Cancel** | discards the changes and exits the dialog |
| **Default** | restores the default configuration |
| **Help** | opens the **Help** of the **Column Configuration** dialog |

## Changing the Columns Sequence

| Step | Action |
|---|---|
| 1 | Select the desired tab in the Data Editor. |
| 2 | Via the context menu open the **Column Configuration** dialog using **Customize Columns**. |
| 3 | Place the cursor on the chosen attribute and move it to the desired location by clicking on the two buttons to the right of the window. |
| 4 | Confirm by clicking **OK**. |

**NOTE:** The first column must always correspond with the attribute **Name**.

# Description of the Data Editor: Variable type selection

## Introduction

The **Data Editor: Variable Type Selection** provides the following options:

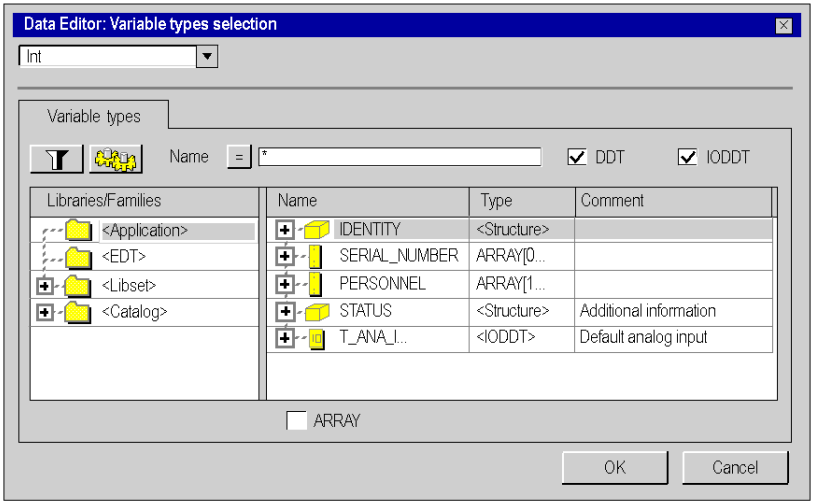
- Selection:
  - of derived data types (typical structures/arrays),
  - the function block data types (EFB\DFB),
- Creating non-typical arrays.

## At a Glance

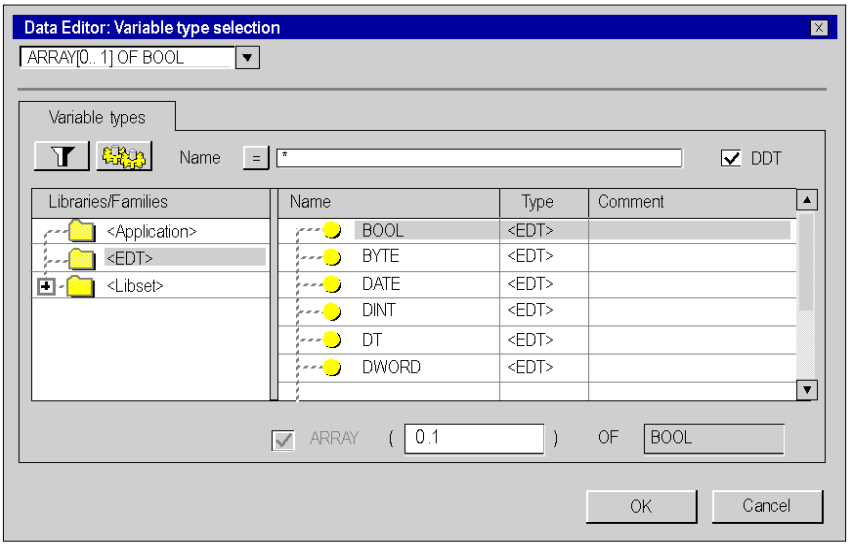
**Data Editor: You can access the variables type selection** by double-clicking on the **Type** column and clicking on  in the Data Editor.

Depending on the tab used to start the editor for the selection of the data type, one of the following windows is displayed:

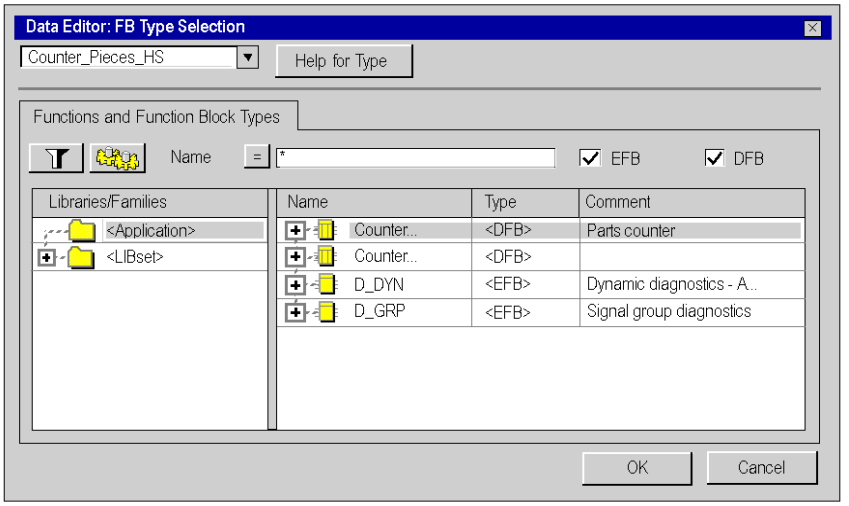
**Variables**:



**DDT Types**:




**Function blocks**:




**NOTE:** Access to the editor for the selection of the data types is not possible via the **DFB types** tab.

### Description of the area Libraries/Families

This area contains directories and subdirectories where various selectable data types are classified.

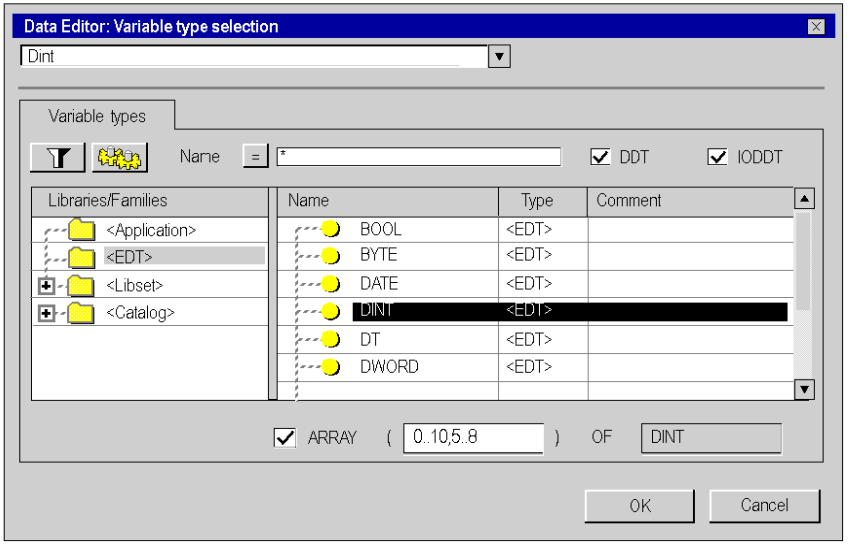
If you choose one of the directories or subdirectories, the types for the right-hand side of the columns Name/Type/Comment are shown.

- The directory **<Application>** shows the data types **used in the project**:
  - the Derived Data Types (DDT); the checkbox **DDT** must be enabled.
  - the Derived Function Block (DFB); the **DFB** checkbox must be enabled.
  - the Derived Data Types that refer to the inputs/outputs (IODDT); the **IODDT** checkbox must be enabled.
  - the data types for the elementary function blocks (EFB); the **EFB** checkbox must be enabled,
  - the Device Derived Data Types (DDDT); the **Device DDT** checkbox must be enabled.
- the **<EDT>** directory shows the types for the elementary data,
- the **<Libset>** directory that is organized into special libraries contains the **families made available by the manufacturer or the functions archived by the** user:
  - the Derived Data Types (DDT); the **DDT** checkbox must be enabled.
  - the data types for the sequence controls (SFC); the **DDT** checkbox must be enabled,
  - the data types for the derived function blocks (DFB); the **DFB** checkbox must be enabled,
  - the elementary function blocks (EFB); the **EFB** checkbox must be enabled,
  - the Device Derived Data Types (DDDT); the **Device DDT** checkbox must be enabled.
- the **<Catalog>** directory that shows the derived data types that reference inputs/outputs (IODDT) **from the manufacturer**; the **IODDT** checkbox must be enabled.

## ARRAY checkbox

If this checkbox is enabled, you can use the editor to select the array types to have a single size or several sizes (maximum six dimensions).

Example: Array with two dimensions 10x4 Dint




## Filtering

You can create filters in the data editor to display the data types according to your requirements.

The **Name** field is used to enter the name of the data type that you want to display in the **Name/Type/Comment** area. You can use place holder symbols (* or ?) to carry out multiple search operations.

Clicking on the button  updates the display corresponding to the filter term defined in the field **Name**.

Clicking on this button  opens the **Filtering data** *(see page 413)* dialog box where you can define various filter attributes.

Clicking on this button  inverts the filter. The button changes from **=** to **<>** and vice versa.

# Filtering of Data

## Introduction

You can create filters in the data editor to display the data according to your requirements.

A filter is the sum of the conditions applied to various attributes.

In every tab, it is standard for all variable instances or data types to be shown.

The various filters you have defined are saved when the data editor is closed and restored when it is opened again.

## At a Glance

The filter parameterization in the data editor is determined depending on the tab chosen. Description of the Data editor different fields depending on the tab:

| Area | In which table is it available ? | Purpose |
|---|---|---|
| Button  | All tabs | Clicking on this button updates the display corresponding to the filter term defined in the **name** field. |
| Button  | All tabs | Clicking on this button opens the **Filtering data** dialog box where you can define the conditions for the individual attributes. |
| Button  | All tabs | Clicking on this button inverts the filter. The button changes from **=** to **<>** and vice versa. |
| **Name** | All tabs | Field where the name (symbol) of the instance or the data type you want to display can be entered. You can use the place holder (* or ?) for this. |
| **EDT** | Variables tab | The enabled check box authorizes the display of the Elementary Data Types (EDT). |
| **EFB** | Function Blocks tab | The enabled check box authorizes the display of the Elementary Function Blocks (EFB). |
| **DDT** | Variables tab | The enabled check box authorizes the display of the Derived Data Types (DDT). |
| **DFB** | Function Blocks tab | The enabled check box authorizes the display of the Derived Function Blocks (DFB). |

| Area | In which table is it available ? | Purpose |
|---|---|---|
| **IODDT** | Variables tab | The enabled check box authorizes the display of the Derived Data Types (DDT) that reference inputs/outputs (IODDT). |
| **Device DDT** | Variables tab | The enabled check box authorizes the display of the Derived Data Types (DDT). |

## The dialog box Filtering data

The dialog box **Filtering data** makes it possible to:

- view the current filter conditions,
- change the current filter conditions,
- define new filter conditions.

**NOTE:** The contents of the **Filtering data** dialog box depends on the tab from which it was called.

## Description of the dialog box columns Filtering data

Description:

| Column | Description |
|---|---|
| **Name** | This column contains the attributes for the (**Variables**, **DDT types**, **Function blocks** or **DFB types**) tab used to open the dialog box. Check the box for the attributes that you want to filter. |
| **Condition** | The column allows you to define the filter condition for every attribute. |
| **Inverted** | Using this checkbox you can invert the filter condition for the attribute in question. |

Options for defining a filter for **Variables**:

| Name | Condition | inverted disabled | inverted enabled |
|---|---|---|---|
| **Comment** | Free text<br>e.g. `*motor*` | All variables whose comment attribute contains the word `Motor` are filtered. | All variables whose comment attribute does not contain the word `Motor` are filtered. |
| **Type** | Free text<br>e.g. `Int` | All variables of data type `Int` are filtered. | All variables not of data type `Int` are filtered. |
| **Address** | Free text<br>e.g. `%MW1` | All variables with the address `%MW1` are filtered. | All variables that do not have the address `%MW1` are filtered. |
| **Global data** | **Yes** | All variables defined as global data are filtered. | All variables not defined as global data are filtered. |
| **Alias Of** | **Yes** | All variables defined as an alias are filtered. | All variables not defined as an alias are filtered. |
| **Value** | **User defined** | All variables defined with an initial value are filtered. | All variables not defined with an initial value are filtered. |

| Name | Condition | inverted disabled | inverted enabled |
|---|---|---|---|
| **Save** | **Accept** | All variables that accept saving and restore operations are filtered. | All variables that do not accept saving and restore operations are filtered. |
| **Constants** | **Yes** | All variables defined as constants are filtered. | All variables not defined as constants are filtered. |
| **Custom** | Free text<br>e.g. `*HMI*` | All variables are filtered if their user-defined text (free text for use in an external tool or for special user requirements) contains the word `HMI`. | All variables whose user-defined text does not contain the word `HMI` are filtered. |
| **Used** | **>0** | All variables used in the project are filtered. | All variables not used in the project are filtered. |
| **Owner** | Free text<br>e.g. `\2.1\*` | All variables which owner begins with `\2.1\` are filtered. | All variables which owner doesn't begin with `\2.1\` are filtered. |
| **HMI Variable** | **Yes** | All variables defined as HMI variable are filtered. | All variables not defined as HMI variable are filtered. |
| **Scanned Variables** | **Yes** | All variables defined as Scanend Variables are filtered. | All variables not defined as Scanned Variables are filtered. |
| **Exchange On STBY** | **Yes** | All variables defined as **Exchange On STBY** Variables are filtered. | All variables not defined as **Exchange On STBY** are filtered. |
| **Task** | Free text<br>e.g. `MAST` | All variables which **Task** attribute is `MAST` are filtered. | All variables which **Task** attribute is not `MAST` are filtered. |
| **Retain** | **Yes** | All variables defined as **Retain** Variables are filtered. | All variables not defined as **Retain** are filtered. |

Options for defining a filter for **DDT types**:

| Name | Condition | inverted disabled | inverted enabled |
|---|---|---|---|
| **Comment** | Free text<br>e.g. `*Motor*` | All DDT types whose comment attribute contains the word `Motor` are filtered. | All DDT types whose comment attribute does not contain the word `Motor` are filtered. |

Options for defining a filter for **Function Blocks**:

| Name | Condition | inverted disabled | inverted enabled |
|---|---|---|---|
| **Comment** | Free text<br>e.g. `*Motor*` | All function blocks whose comment attribute contains the word `Motor` are filtered. | All function blocks whose comment attribute does not contains the word `Motor` are filtered. |
| **Diag.** | **Accept** | All function blocks that accept the attribute `Diag.` are filtered. | All function blocks that do not accept the attribute `Diag.` are filtered. |
| **Type** | Free text<br>e.g. `TON` | All function blocks of type `TON` are filtered. | All function blocks not of type `TON` are filtered. |
| **Custom** | Free text<br>e.g. `*HMI*` | All function blocks are filtered if their user-defined text (free text for use in an external tool or for special user requirements) contains the word `HMI`. | All function blocks whose user-defined text does not contain the word `HMI` are filtered. |
| **Used** | **>0** | All function blocks used in the project are filtered. | All function blocks not used in the project are filtered. |
| **Owner** | Free text<br>e.g. `\2.1\*` | All function blocks which owner begins with `\2.1\` are filtered. | All function blocks which owner doesn't begin with `\2.1\` are filtered. |
| **HMI Variable** | **Yes** | All function blocks defined as HMI variable are filtered. | All function blocks not defined as HMI variable are filtered. |
| **Exchange On STBY** | **Yes** | All function blocks defined as **Exchange On STBY** Variables are filtered. | All function blocks not defined as **Exchange On STBY** are filtered. |
| **Task** | Free text<br>e.g. `MAST` | All function blocks which **Task** attribute is `MAST` are filtered. | All function blocks which **Task** attribute is not `MAST` are filtered. |
| **Retain** | **Yes** | All function blocks defined as **Retain** Variables are filtered. | All function blocks not defined as **Retain** are filtered. |

Options for defining a filter for **DFB types**:

| Name | Condition | inverted disabled | inverted enabled |
|---|---|---|---|
| **Comment** | Free text<br>e.g. `*Motor*` | All DFB types whose comment attribute contains the word `Motor` are filtered. | All DFB types whose comment attribute does not contain the word `Motor` are filtered. |
| **Diag.** | **Accept** | All DFB types that accept the attribute `Diag.` are filtered. | All DFB types that do not accept the attribute `Diag.` are filtered. |
| **Protection** | **Protected** | All protected DFB types are filtered. | All non-protected DFB types are filtered. |

# Exporting a subset of variables

## Introduction

The chapter *Import / Export, page 1719* explains how to export variables.

However, all variables and FB instances, and subsets, were always exported (EDT, DDT, IODDT, EFB, DFB).

In addition, a subset of variables may be filtered out and/or selected via the Data Editor and only this subset will then be exported.

## Filter

Filter the variables as described in the chapter *Filtering of Data, page 412*.

## Selecting

Click on a variable to select it in the Data Editor. You can select multiple variables by holding down the **Alt** or **Ctrl** keys.

## Export

Export the filtered or selected variables via the context menu (right-click).

# Analyze DDT and DFB Data Types

## At a Glance

DDT (Derived Data Types) and DFB (Derived Function Block) will have one of the following status **after setup**:

- Status "in Process",
- Status "after the analysis".

This means that the data are instantiated with a type that is in process. The created instance has characteristics of the prior type that was analyzed prior to processing.

After the analysis, to the extent that the type has no errors, the applicable instances within the application will be updated..

## Local analysis of data

Running the local analysis checks the selected data type for compatibility with the corresponding instances throughout the entire project.

For DFB types, each section of the type is analyzed separately..

Perform the following steps to start a local analysis:

| Step | Action |
|---|---|
| 1 | Select the desired data type in the **DDT Types** or **DFB Types** tab. |
| 2 | Select the **Analyze Type**command from the shortcut menu. |
| 3 | Check the result of the analysis in the **Display Window** on the lower part of the window. |

**NOTE:** Double-click on an error row in the display window to go to the source of the error.

## Global analysis of data

Running the global analysis checks all selected data types for compatibility with the corresponding instances throughout the entire project.

Perform the following steps to start a global analysis:

| Step | Action |
|---|---|
| 1 | Select the **Generate> Analyze Project** command in the menu. |
| 2 | Check the result of the analysis in the display window on the lower part of the window. |

**NOTE:** Double-click on an error row in the display window to go to the source of the error.

# Purge of Unused Instances

## Overview

To purge unused instances of EDTs, EFBs and DFBs the Data Editor tabs provide the respective context menus.

## Purge Unused Variable

The purge depends on the check boxes (EDT, DDT, IODDT and Device DDT), i.e. only the visible data (of checked types) will be purged.

| Step | Action |
|---|---|
| 1 | Select the **Variables** tab of the Data Editor. |
| 2 | Right-click any column of the tab. |
| 3 | Select the menu command **Purge Unused Variables** from the context menu. |
| 4 | Confirm with **Yes**.<br>**Result**: The unused data will be purged. |

## Purge DDT Types

**NOTE:** The purge feature is not available for DDT types.

## Purge Unused FB Instances

The purge depends on the check boxes (EFB, DFB), i.e. only the visible data (of checked types) will be purged.

| Step | Action |
|---|---|
| 1 | Select the **Function Blocks** tab of the Data Editor. |
| 2 | Right-click any column of the tab. |
| 3 | Select the menu command **Purge Unused FB Instances** from the context menu. |
| 4 | Confirm with **Yes**.<br>**Result**: The unused data will be purged. |

**NOTE:** When a FFB is not available for a kind of PLC (after a section import for instance) but still in the application (removed from the section but still in the list of the Application FFBs), the purge feature should be used before building the application.

## Purge Unused Private Variables/DFB Types

From the context menu of the **DFB Types** tab two purge operations are accessible.

| Step | Action |
|---|---|
| 1 | Select the **DFB Types** tab of the Data Editor. |
| 2 | Right-click any column of the tab. |
| 3 | Select one of the two menu commands **Purge Unused Private Data Instances** or **Purge unused Types** from the context menu. |
| 4 | Confirm with **Yes**.<br>**Result**: The unused data will be purged. |

# Local printing of data

## Introduction

Printing is a user-defined process and is executed using the following items:

- selected tab,
- Filtering variables,
- Sorting variables,
- Using certain variables

## How to print

Carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the user-defined adjustment of the printing process in the Data Editor:<br>● Select the tab,<br>● Select the filter *(see page 412)*:<br>● Select the sorting order (alphabetically ascending or descending),<br>● Using structured variables |
| 2 | Select the **File** → **Print** menu command<br>or<br>use the key combination **Ctrl+P**<br>or<br>use the toolbar icon  . |

**NOTE:** Local printing takes the access security into consideration. Only information appearing in the window is printed.

**NOTE:** If an IODDT, a DDT or an Array element of an instance has an alias, the instance is displayed with the alias of the element as in the **Data Editor**.

# Save the context of the Data Editor

## Principles

When you leave the Data Editor, the configuration of the editor (visual aspect) will be saved. This includes:

- the size and position of the Editor window,
- the selected tab,
- the filter of the variables display,
- The configuration of the columns in each tab.

The following information will not be saved:

- An instance or variable type will be hidden when the Editor is reopened.
- The selection of single or multiple variables.

# Part III
# Communication

## Subject of this Part

This part describes the communication editors.

## What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 15 | Presentation of the communication editors | 425 |
| 16 | Network Configuration | 427 |
| 17 | Configuration of X-Way Routing Premium Stations | 435 |

# Chapter 15
# Presentation of the communication editors

## Presentation of the communication editors of the project browser

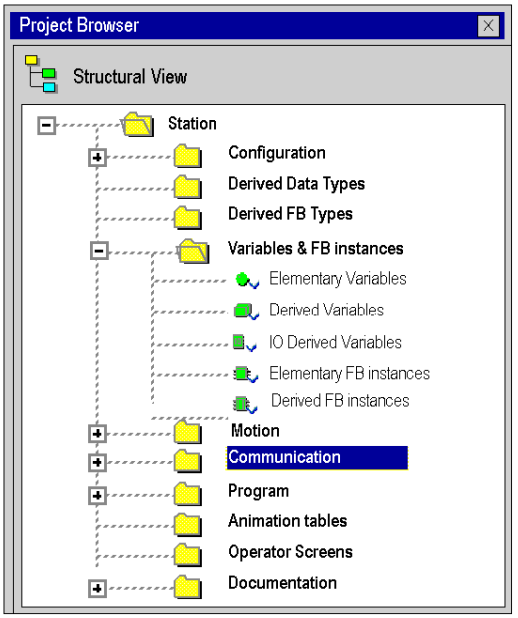
### At a Glance

The communication editors allow you to configure and manage the different communication entities at project level.

They are accessible through the project browser by clicking the **Communication** tab.

### Illustration

The figure below shows an example of the project browser



# Chapter 16
# Network Configuration

## Subject of this Chapter

This chapter presents the tools for configuring a network at the global level and at the station level.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Network Configuration Principle Using Unity Pro | 428 |
| Creating a Logic Network | 429 |
| Configuring a Logic Network | 431 |
| Associating a Logic Network with Network Hardware | 432 |

# Network Configuration Principle Using Unity Pro

## At a Glance

With Unity Pro, the installation of a network takes place from the application browser and from the hardware configuration editor.

The method involves the following four steps:

- creation of a logic network,
- configuration of the logic network,
- declaration of the module or of the PCMCIA card (for Premium),
- association of the card or of the module with the logic network.

These four methods are presented further on in this documentation.

**NOTE:** The advantage of this method is that from the second step onwards, you can design your communication application (you do not need the hardware to start working) and use the simulator for functional testing of it.

**NOTE:** The first two steps are carried out in the project browser and the following two in the hardware configuration editor.

This manual introduces the method. For details of the various network configurations, please refer to the following documentation:

- Ethernet configuration for Premium *(see Premium and Atrium Using Unity Pro, Ethernet Network Modules, User Manual)*, Ethernet configuration for M340 *(see Modicon M340 for Ethernet, Communications Modules and Processors, User Manual)*, and Ethernet configuration for Modicon M580,
- Modbus Plus configuration *(see Premium and Atrium Using Unity Pro, Modbus Plus Network, User Manual)*,
- Fipway configuration *(see Premium and Atrium using Unity Pro, Fipway Network, User Manual)*.

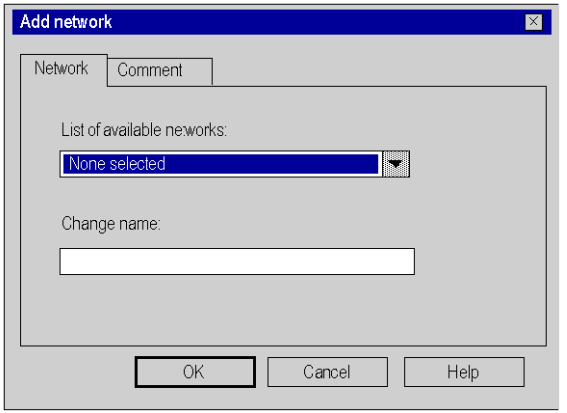
# Creating a Logic Network

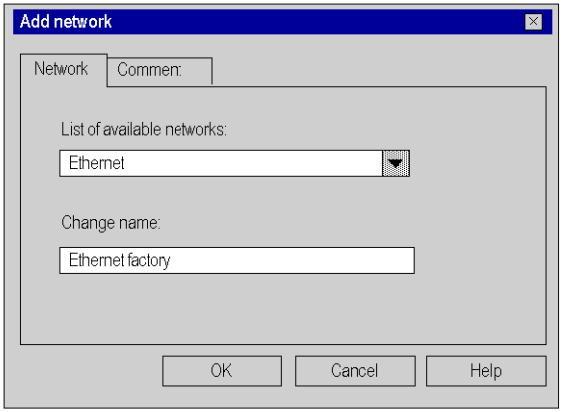
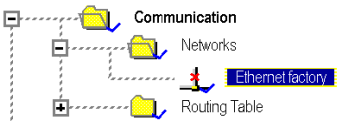
## At a Glance

The first step in implementing a communication network is to create a logic network.

## Creating a Logic Network

The following table describes how to create a network using the project browser.

| Step | Action |
|---|---|
| 1 | Expand the *Communication* directory in the project browser.<br>**Result:**:<br>  |
| 2 | Right-click in the *Networks* sub-directory and select the **New network** option.<br>**Result:**:<br>  |

| Step | Action |
|---|---|
| 3 | Select the network that you want to create from the list of available networks and give it a meaningful name.<br>**Result:** Example of an Ethernet network:<br>  <br>**Note**: You can also add a comment, if you so desire, by clicking on the **Comment** tab. |
| 4 | Click OK and a new logic network is created.<br>**Result:** We have just created the Ethernet network that appears in the project browser<br>  <br>**Note**: As you can see, a small icon indicates that the logic network is not associated with any PLC hardware. Furthermore, the small blue "v" sign indicates that the project needs to be rebuilt before it can be used in the PLC. |

# Configuring a Logic Network

## At a Glance

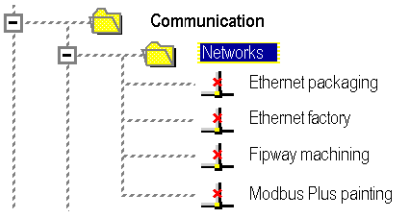
The second step in implementing a communication network is to configure a logic network.

This manual introduces the access to network configuration. For information on how to configure the various networks, please refer to the following documentation:

- Ethernet configuration for Premium *(see Premium and Atrium Using Unity Pro, Ethernet Network Modules, User Manual)*, Ethernet configuration for M340 *(see Modicon M340 for Ethernet, Communications Modules and Processors, User Manual)*, and Ethernet configuration for Modicon M580,
- Modbus Plus configuration *(see Premium and Atrium Using Unity Pro, Modbus Plus Network, User Manual)*,
- Fipway configuration *(see Premium and Atrium using Unity Pro, Fipway Network, User Manual)*.

## Configuring a Logic Network

The table below describes how to access the configuration of a network from the project browser.

| Step | Action |
|---|---|
| 1 | In the project browser, expand the directory tree under the **Networks** sub-tab located in the **Communication** tab of the tree directory to display all the project networks.<br>**Example:**:<br>  |
| 2 | Double-click the network you want to configure to obtain the network configuration window.<br>**Note**: The windows differ according to the network family selected. However, for all networks, from this window it is possible to configure the Global Data, IPO scanning, Peer Cop utilities, common words, etc.<br>**Note**: For Ethernet networks, an intermediate step is necessary, which involves selecting the family of the module that will be used in the hardware configuration. |

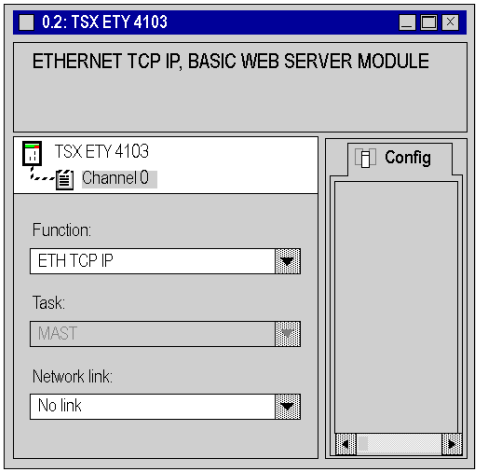
# Associating a Logic Network with Network Hardware

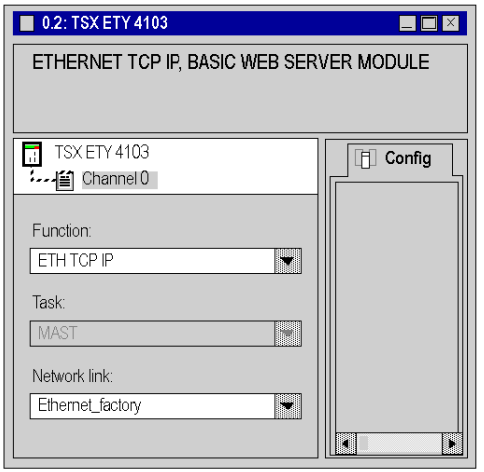
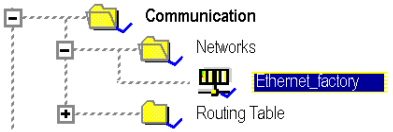
## At a Glance

The final step in implementing a communication network is to associate a logic network with a network module, Modbus Plus card or Fipway card. Although the screens differ, the procedure is the same for each network device.

## How to Associate a Logic Network

The following table describes how to associate a logic network to a network device declared in the hardware configuration editor.

| Step | Action |
|---|---|
| 1 | Open the hardware configuration editor. |
| 2 | Right-click the device (Ethernet module, Fipway PCMCIA card or Modbus Plus PCMCIA card) that you wish to associate with a logical network. |
| 3 | Select the channel and function.<br>**Result:** For a TSX ETY 4103 module:<br>  |

| Step | Action |
|---|---|
| 4 | In the **Network link** field, select the network to be associated with the card.<br>**Result:**<br>  |
| 5 | Confirm your choice and close the window.<br>**Result:** The logic network is associated with the device. The icon associated with this logic network changes and indicates the existence of a link with a PLC. Furthermore, the rack, module and channel numbers are updated in the logic network configuration screen. In our example we obtain the following project browser:<br>  |

# Chapter 17
# Configuration of X-Way Routing Premium Stations

## Subject of this Chapter

This chapter presents the operating modes required for configuring X-Way routing Premium stations.

## What Is in This Chapter?

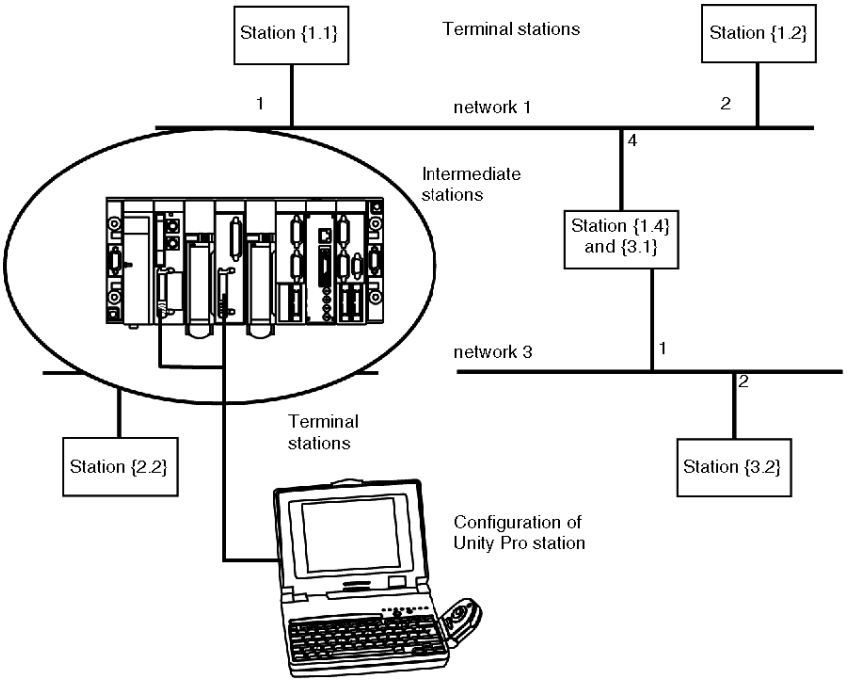
This chapter contains the following topics:


| Topic | Page |
|---|---|
| Configuration | 436 |
| Configuration of Multi-Network Services | 437 |
| Configuring an X-Way Router Module | 439 |
| Examples of X-Way Routing Stations | 443 |
| Examples of Partial Routing | 447 |

# Configuration

## At a Glance

In an intermediate station, the management of several network couplers requires a configuration phase in order to distribute the functional characteristics to the various network entities.




**NOTE:** Multi-network routing information is constructed at the station level at the time of configuration of each bridge. No consistency check is done on routing data for the same network architecture.

# Configuration of Multi-Network Services

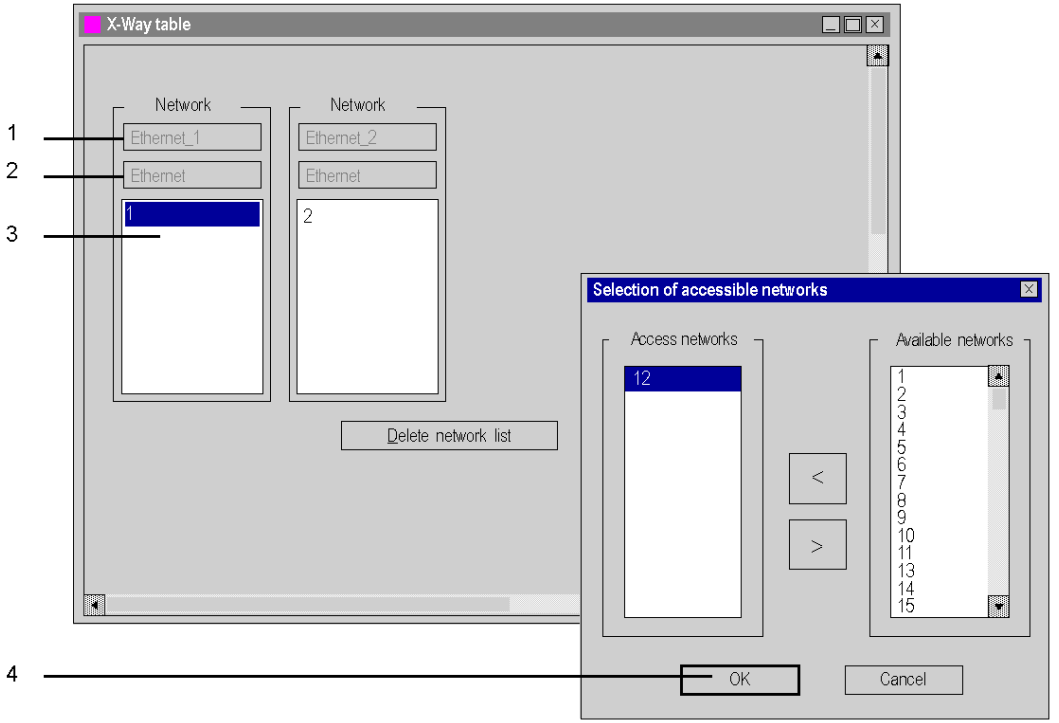
## At a Glance

In a station that supports various network modules, each network connection point is considered as an address for the station. When configuring each module, it is necessary to define the list of network numbers that are accessible for each connection point.

Depending on the processor selected during hardware configuration, a bridge station can only manage 3 or 4 network modules. The table will therefore have a maximum of 4 elements.

## Illustration

A specific screen allows entry of routing data for all the network modules of a station.



## Elements and Functions

The following table describes the various zones in the configuration screen:

| Label | Field | Function |
|---|---|---|
| 1 | Logical network | Used to display the logical network name. |
| 2 | Network type | Used to display the network type. |
| 3 | Accessible networks | Used:<br>● for the unshaded **Logical Network** zone, to enter the list of networks accessible by this module,<br>● for the shaded **Logical Network** zones, to display the list of networks accessible by these modules. |
| 4 | Available networks | Used to select the networks accessible by a module configured as a bridge.<br>A list of numbers from 1 to 127 shows the networks available for a connection point. Each network number selected as being accessible is removed from the list of available networks in order to avoid configuration errors. |

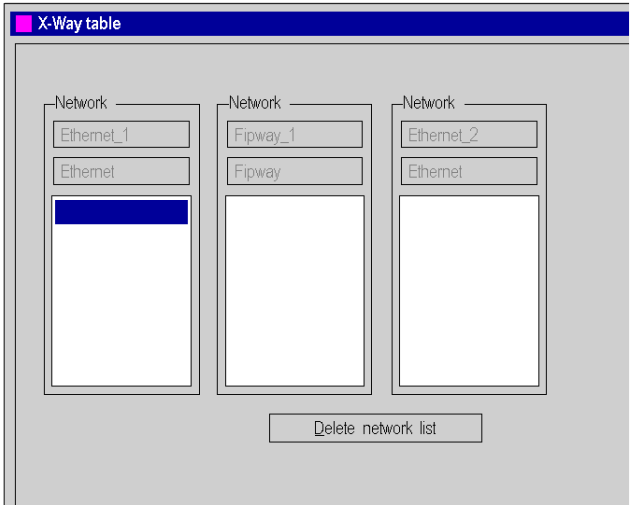
# Configuring an X-Way Router Module

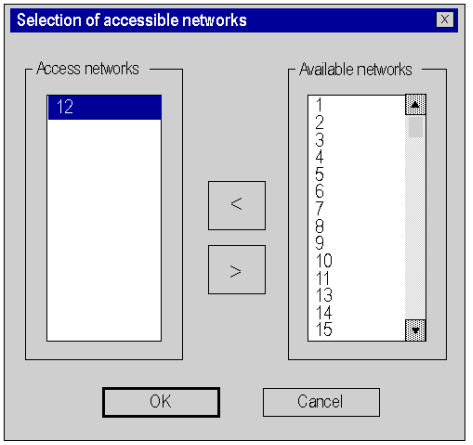
## At a Glance

Before configuring the module as an X-Way router, the station's logical networks must be created.

## Procedure

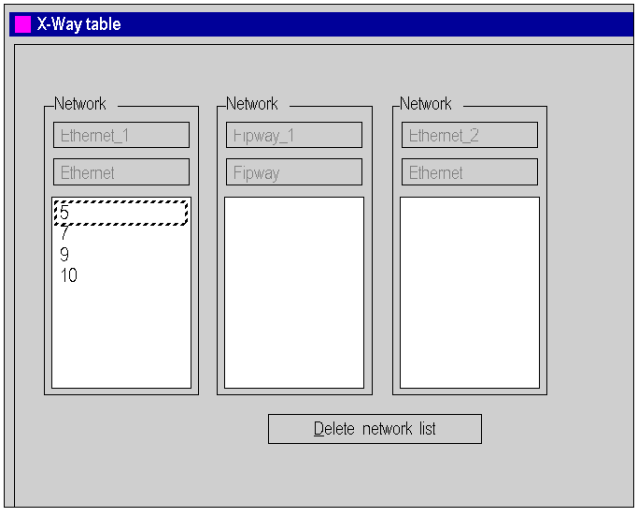
The following procedure is used to access, and then configure the station's module as an X-Way router.

| Step | Action |
|---|---|
| 1 | Open the **Communication** tab in your project browser and in the **Routing table** tab, click the **X-Way table** tab.<br>**Result**: The following window appears.<br>  <br>If the list of accessible module networks is empty, the window appears automatically (without double-clicking). |

| Step | Action |
|---|---|
| 2 | Double-click the highlighted field in order to configure the first network.<br>**Result:** The **Selection of accessible networks** window appears.<br>  |
| 3 | Double-click the number of the required network from the **Available networks** scroll list.<br>**Result:** The network number is assigned in the **Access networks** scroll list. |
| 4 | Perform operation 3 as many times as necessary to define all the networks accessible by the module.<br>Once finished, proceed to step 5. |
| 5 | Confirm the selection by clicking **OK**. |
| 6 | Confirm the configuration of the X-Way router by closing the window or clicking the **Enable** button in the toolbar. |

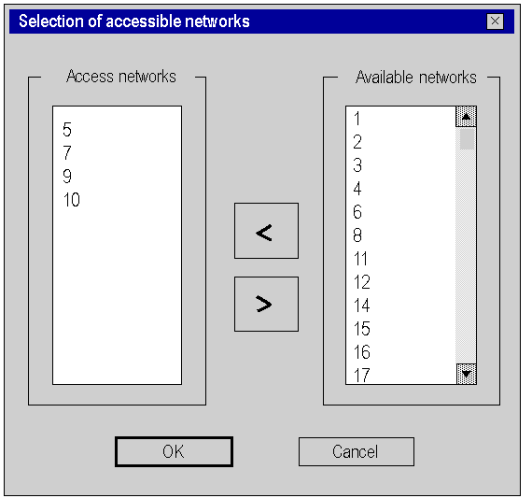
## Removing the Bridge Function

It is possible to remove the bridge function from the module.

| Step | Action |
|---|---|
| 1 | Access the following X-Way window.<br>  |
| 2 | Click **Delete network list**. |
| 3 | Confirm the configuration |

## Removing Access to a Network

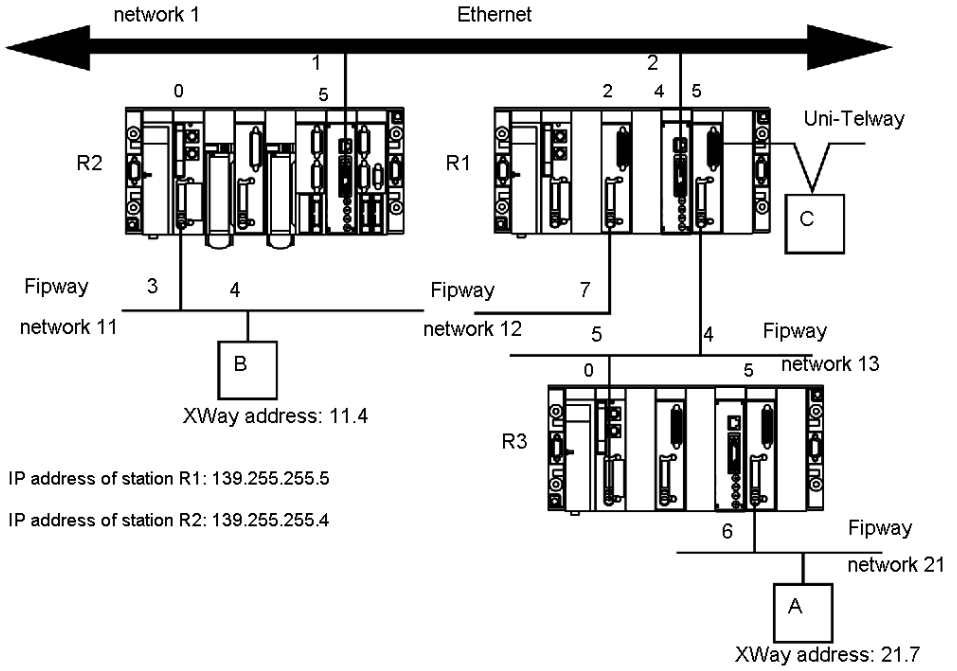
It is possible to remove access to a single network.

| Step | Action |
|---|---|
| 1 | Access the following X-Way window of the accessible networks.<br>  |
| 2 | Double-click the numbers of the accessible networks to be removed (left column).<br>**Result:** The network number is reassigned in the **Available networks** scroll list. |
| 3 | Confirm the selection by clicking **OK**. |
| 4 | Confirm the router configuration. |

# Examples of X-Way Routing Stations

## At a Glance

Each station must be configured in order to define the list of accessible networks.



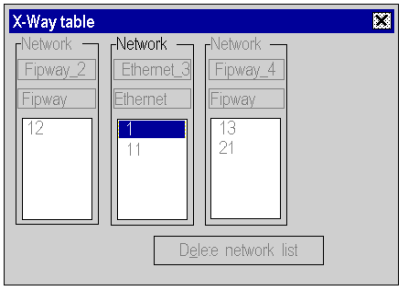
## Configuration of Station R1

The module at slot 2 can only access network 12.

The module at slot 4 can only access network 1 and 11.

The module at slot 5 can access networks 13 and 21.

The bridge configuration of the station is therefore as follows:

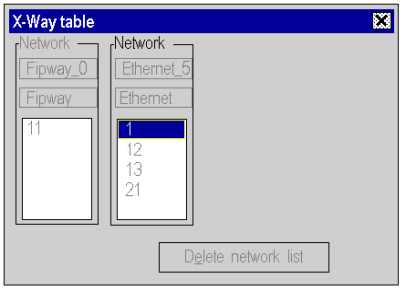



## Configuration of Station R2

The module at slot 0 can only access network 11.

The module at slot 5 can access networks 1, 12, 13 and 21.

The bridge configuration of the station is therefore as follows:

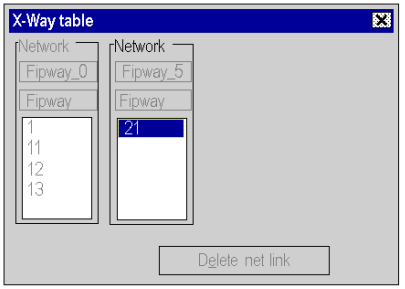


## Configuration of Station R3

The module at slot 0 can access networks 13, 12, 1 and 11.

The module at slot 5 can only access network 21.

The bridge configuration of the station is therefore as follows:



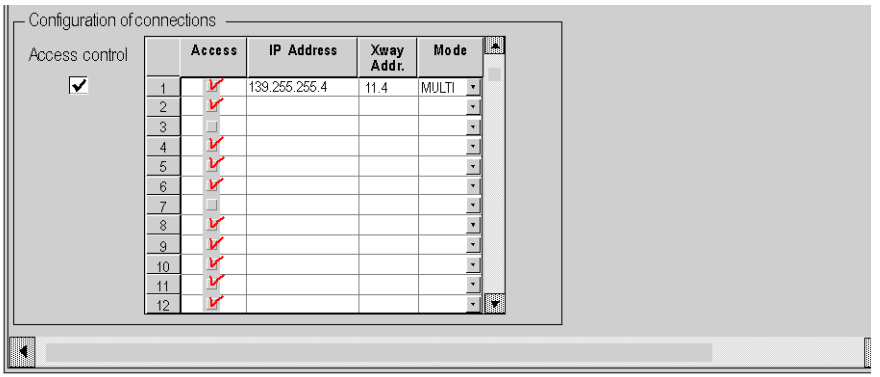

## Messaging

To use the communication function *(see Unity Pro, Communication, Block Library)* such as Read_VAR for ethernet exchange between stations, configure the TCP/IP Messaging in the Ethernet network configuration *(see Premium and Atrium Using Unity Pro, Ethernet Network Modules, User Manual)* screen. In the **Messaging** tab, for each exchange set the IP address and target XWay address.

For example if station R3 needs to communicate with station B, in R1**Messaging** tab:

- set the R2 IP address (139.255.255.4) in the field **IP address**, and
- set the B XWay address (11.4) in the field **XWay address**.

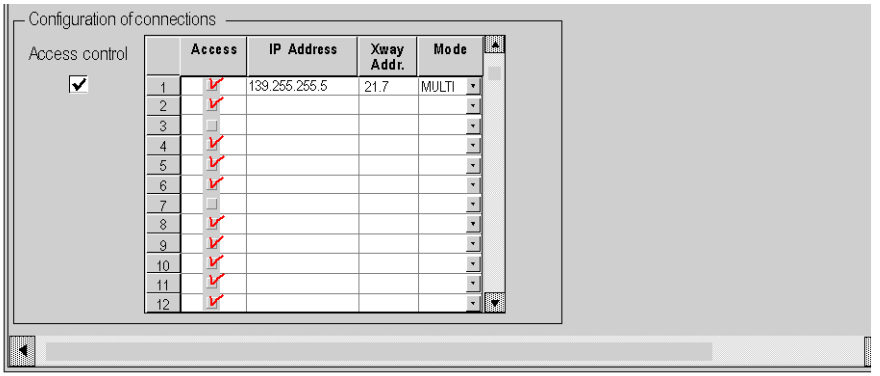
The following illustration shows the R1 **Messaging** tab:




For another example if station B needs to communicate with station A, in R2, **Messaging** tab:

- set the R1 IP address (139.255.255.5) in the field **IP address**, and
- set the A XWay address (21.7) in the field **XWay address**.

The following illustration shows the R2 **Messaging** tab:



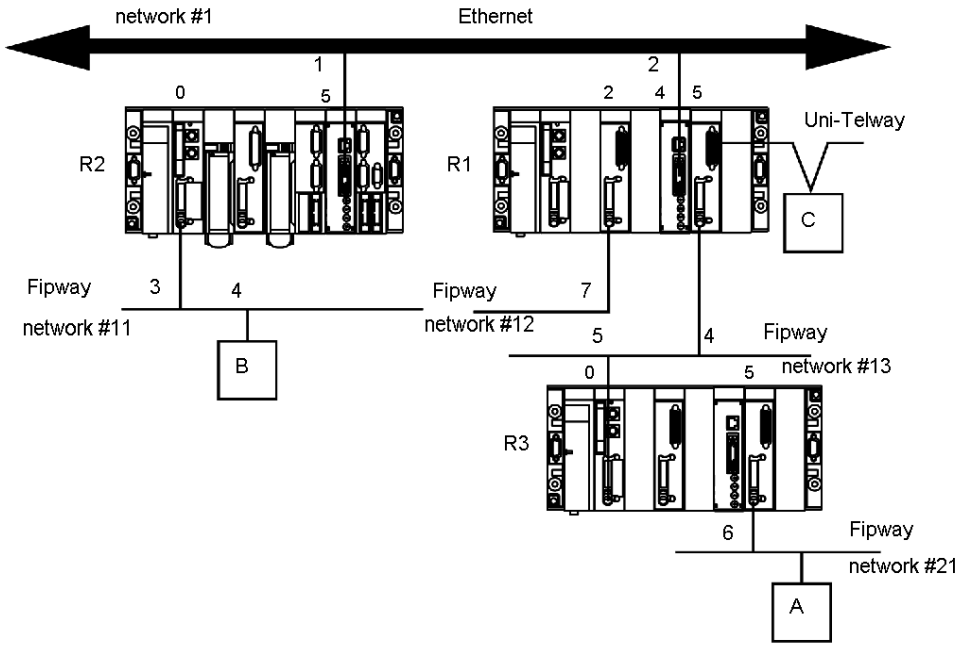
# Examples of Partial Routing

## At a Glance

When configuring a module as a bridge, it is possible to assign to it only a part of the available networks, instead of all of them. This selection is used to define a partial routing.

## Illustration

Each station must be configured in order to define the list of accessible networks.



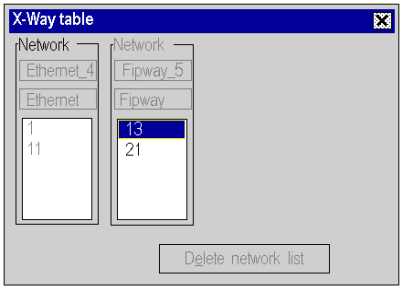
## Configuration of Station R1

The module at slot 2 is not involved in the routing of data.

The module at slot 4 can access networks #1 and #11.

The module at slot 5 can access networks #13 and #21.

The bridge configuration of the station is therefore as follows:

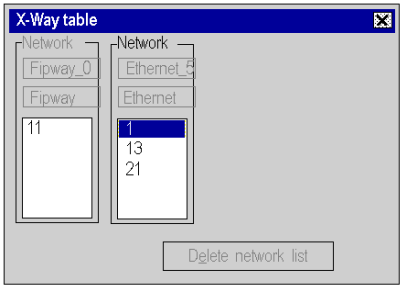



## Configuration of Station R2

The module at slot 0 can only access network #11.

The module at slot 5 can access networks #1, #13 and #21. Network #12 is inaccessible.

The bridge configuration of the station is therefore as follows:

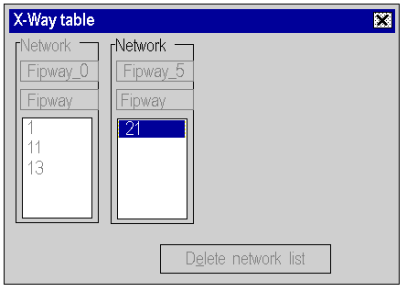


## Configuration of Station R3

The module at slot 0 can access networks #13, #1 and #11. Network #12 is no longer accessible.

The module at slot 5 can only access network #21.

The bridge configuration of the station is therefore as follows:



# Part IV
# Programming

## Subject of this part

This part describes the programming editors.

## What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 18 | Accessing Tasks and Sections | 453 |
| 19 | Unity Pro Software Options | 515 |
| 20 | Common Function for graphical language | 575 |
| 21 | FBD Editor | 611 |
| 22 | LD Editor | 709 |
| 23 | SFC Editor | 861 |
| 24 | Common Function for Textual Language | 1007 |
| 25 | IL Editor | 1041 |
| 26 | ST Editor | 1073 |
| 27 | LL984 Editor | 1111 |
| 28 | Programming Languages Data Selection Dialog Box | 1181 |
| 29 | I/O Objects tab | 1201 |
| 30 | Memory Tab | 1219 |
| 31 | Type Library Browser | 1225 |
| 32 | Search/Replace Tool | 1235 |
| 33 | User's Function Blocks | 1251 |

# Chapter 18
# Accessing Tasks and Sections

## Overview

This section describes how to access tasks and sections.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 18.1 | Access to Tasks | 454 |
| 18.2 | Accessing Sections | 457 |

# Section 18.1
## Access to Tasks

### Subject of this Section

This section covers the creation and modification of program tasks.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Create and Configuration of a Task | 455 |
| Displaying and Modifying Task Properties | 456 |

# Create and Configuration of a Task

## General

The first operation you need to perform to create an application program is to define the tasks.

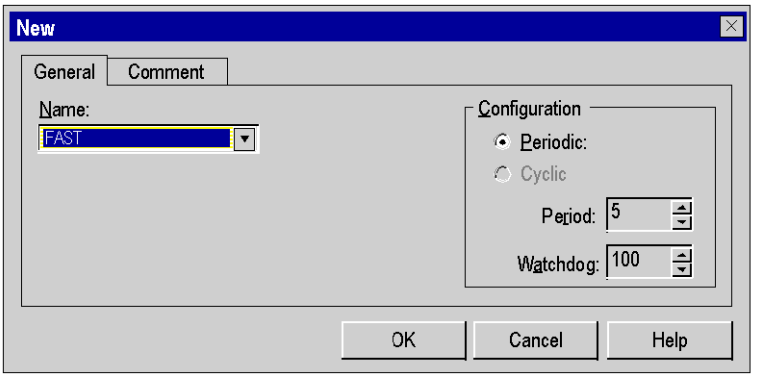
By default, only the master task is proposed. It is possible to create the following tasks:

the FAST task and the auxiliary tasks AUX0 to 3.

**NOTE:** For a fipio *(see Premium and Atrium Using Unity Pro, Fipio Bus, Setup Manual)* configuration, the window displayed is different.

## Creating a Task

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the project browser, double-click on the program directory.<br>The MAST directory appears in the **Tasks** directory. |
| 2 | Right-click on the Tasks directory, and then execute the **New task...** command from the contextual menu. |
| 3 | Click on **New task...**, and the following dialog box appears:<br>  |
| 4 | Select the task:<br>● FAST: Fast task<br>● AUX 0 , AUX1, AUX2 or AUX3: Auxiliary tasks (for processors with auxiliary tasks) |
| 5 | Select the type of scanning:<br>● Periodic<br>● or cyclic (for master task only) |
| 6 | Set the task period. |
| 7 | Set the **Watchdog** value which must be greater than the period value. |
| 8 | The **Comment** tab can be used to add a comment to describe the role of the task for example. |

# Displaying and Modifying Task Properties

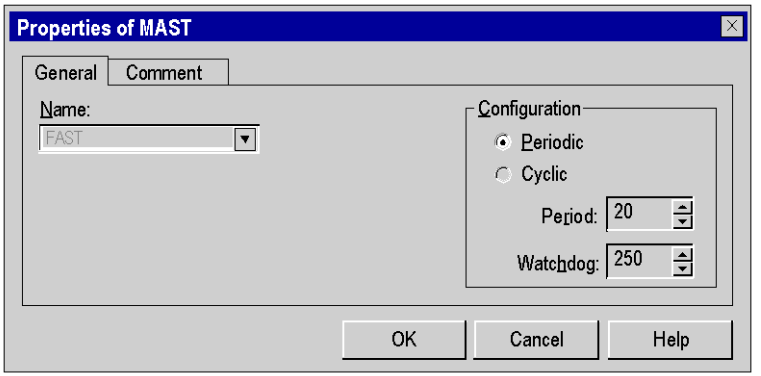
## General

This procedure should be followed to display and/or modify the properties of an existing task.

**NOTE:** For a fipio *(see Premium and Atrium Using Unity Pro, Fipio Bus, Setup Manual)* configuration, the window displayed is different.

## Task Properties

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the project browser, double-click on the program directory.<br>The MAST directory (as well as the FAST and AUX directories if they have been created) appears in the **Tasks** directory. |
| 2 | Right-click on the MAST, FAST, or AUX directory, and then execute the **Properties** command from the contextual menu. |
| 3 | Click on **Properties**, and the following dialog box appears:<br>  |
| 4 | Select the type of task:<br>● Periodic<br>● or cyclic |
| 5 | If you have selected a periodic task, set the task period. |
| 6 | Set the **Watchdog** value which must be greater than the period value. |
| 7 | The **Comment** tab can be used to add a comment to describe the role of the task for example. |

# Section 18.2
## Accessing Sections

### Overview

This section describes accessing sections.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Creating an FBD, LD, IL, or ST Section | 458 |
| Properties Dialog Box for FBD, LD, IL, or ST Sections | 461 |
| Creating SFC sections | 467 |
| Properties Dialog Box for SFC Sections | 470 |
| Creating action, transition and macro sections | 477 |
| Action, transition and macro sections properties dialog box | 478 |
| Create a Sub-Program Section (SR) | 481 |
| Property Dialog of the Sub-Program Sections | 483 |
| Creating a Timer Event Section | 486 |
| Properties dialog box for Timer Event Sections | 489 |
| Create an I/O Event Section | 493 |
| Property Dialog of I/O Event Sections | 496 |
| Opening a Section | 500 |
| Deleting sections | 501 |
| Execution order of sections | 502 |
| Creating LL984 Segments and Networks | 503 |
| Modifying LL984 Segment Properties | 508 |
| Modifying LL984 Network Properties | 510 |
| Scheduling LL984 Segments and Network | 512 |
| Deleting, Cutting, Copying and Pasting LL984 Segments and Network | 513 |

# Creating an FBD, LD, IL, or ST Section

## Introduction

Create FBD, LD, IL, or ST sections for Modicon M340:
- in the MAST and FAST tasks
- in offline and online mode

Create FBD, LD, IL, or ST sections for Modicon M580:
- in the MAST, FAST, AUX0, and AUX1 tasks
- in offline and online mode

Create FBD, LD, IL, or ST sections for Quantum, Premium and Atrium:
- in MAST, FAST, and AUX tasks
- in offline and online mode

## Information for creating LD sections

An LD section contains 11-64 columns and 17-2000 rows.

The number of columns is set in the **Tools** → **Project Settings** dialog in the **Editors** tab in the **Ladder Daigram (LD)** area at **Number of columns**.

This setting only applies to sections to be created, not existing sections. Verify that you define the number of columns before you create a new LD section.

(The number of rows and columns can be changed during the creation of section contents.)

## Information for creating FBD sections

An FBD section contains 86,400 grids (number of rows x number of columns).

The number of rows and columns is set in the **Tools** → **Project Settings** dialog in the **Editors** tab in the **Function Block Diagram (FBD)** area.

This setting only applies to sections to be created, not existing sections. Always define the number of rows and columns before you create a new FBD section.

(The number of rows and columns can be changed during the creation of section contents.)

## Creating a section

Creating a section:

| Step | Action |
|---|---|
| 1 | Left-click the **Sections** directory of the desired task, or place the cursor using the arrow keys on this directory. |
| 2 | Click the right mouse button or use **Shift**+**F10** to select **New Section** in the context menu.<br>**Result:** The properties dialog box *(see page 461)* for the section is opened. |
| 3 | Enter the section name.<br>The section name must be unique in the entire project and conform to the general naming conventions. |
| 4 | Select the programming language of the section. |
| 5 | Select the type of access protection if necessary from the **Protection** list box:<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also section Project Protection *(see page 186)*. |
| 6 | Select the functional module to which the section is allocated from the **Localization** tab if necessary.<br>See also section Functional module *(see page 221)*. |
| 7 | Define a Boolean variable in the **Condition** tab with which the section can be activated/deactivated (section active when variable=1) if required.<br>You have the following options to select variables:<br>● You can enter the name of the variable/address directly or copy it from the clipboard.<br>● Use the command button **...**to open a variable selection dialog box *(see page 330)*.<br>Activate the **Can be forced** check box when you want to force the variable. |
| 8 | **Only applies to Quantum projects:**<br>Assign a read RIO drop and a write RIO drop to the section in the **Remote IO** tab if required.<br>See also section Executing Quantum sections with distributed inputs/outputs *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 9 | Enter a comment in the **Comment** tab if desired. |
| 10 | If you wish to create additional sections, confirm the entries with **Apply**. In this case, the Property dialog box will remain open and you can generate additional sections.<br>If you do not wish to create additional sections, confirm the entries with **OK**. In this case, the Property dialog box will close and the generated section will open automatically. |

## Displaying or modifying section properties

Displaying or modifying section properties:

| Step | Action |
|---|---|
| 1 | Left-click on the desired section, or place the cursor on this section with the arrow keys. |
| 2 | Select (right-click or **Shift+F10** key combination) the **Properties... menu command from the shortcut menu.**.<br>**Result:** The properties dialog box *(see page 461)* for the section is opened. |
| 3 | Implement the desired changes. |
| 4 | Confirm all entries with **OK**. |

## Importing/exporting sections

See:

- Importing programs *(see page 1735)*
- Exporting programs *(see page 1733)*

# Properties Dialog Box for FBD, LD, IL, or ST Sections

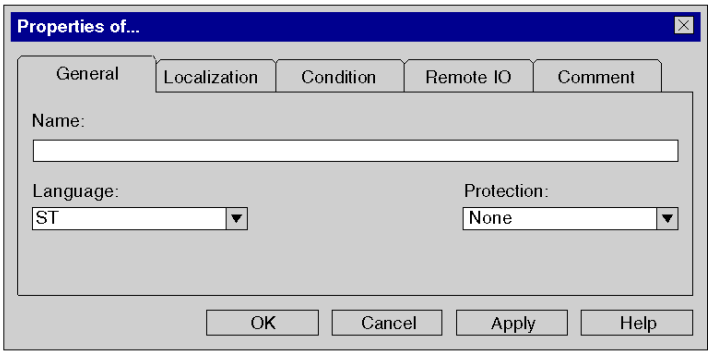
## Structure of the properties dialog box

The properties dialog box consists of four or five tabs depending on the platform of the project:

- **General**
  Enter the name, the programming language and the type of access protection of the section in this tab.
- **Localization**
  You can enter the functional module of the section in this tab.
- **Condition**
  You can enter a Boolean variable which can enable/disable the section in this tab.
- **Remote IO**
  This tab is only available for Quantum projects.
  Assign a read RIO drop and a write RIO drop to the section in this tab.
  Assigning drops will remove them from the TASK I/O update routine and will be managed from the section.
  When disabling a section by a condition, the read and write of the assigned drops are still active.
- **Comment**
  You can enter a comment in this tab.

## General tab

Representation of the **General** tab:

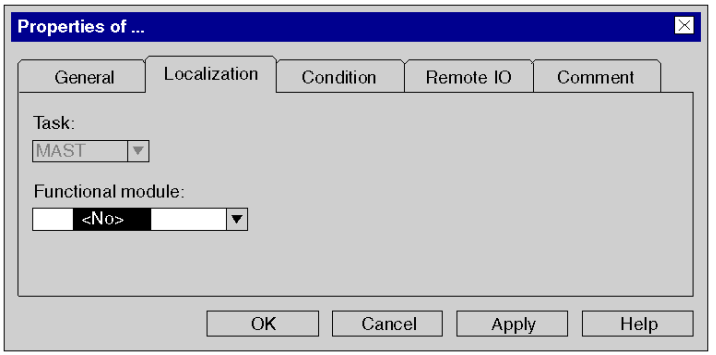


Elements of the **General** tab:

| Element | Description |
|---|---|
| **Name** | Enter the name of the section in this text box.<br>The section name must be unique in the entire project and conform to the general naming conventions. |
| **Language** | Select the programming language of the section from this list box.<br>**Note:** The programming language can only be defined for new sections. The programming language cannot be changed for existing sections. |
| **Protection** | Select the type of access protection from this list box.<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also *Section and Subroutine Protection (see page 186)*. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections. The generated section will not open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Localization tab

Representation of the **Localization** tab:

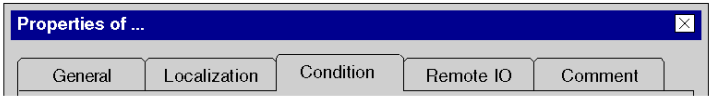



Elements of the **Localization** tab:

| Element | Description |
|---|---|
| **Task** | The task of the section is displayed in this field. |
| **Functional module** | Select the functional module to which the section is to be allocated to in this list box.<br>For detail, refer to chapter *Functional Modules* *(see page 221)*. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections. The generated section will not open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Condition tab

Display of the **Condition** tab:

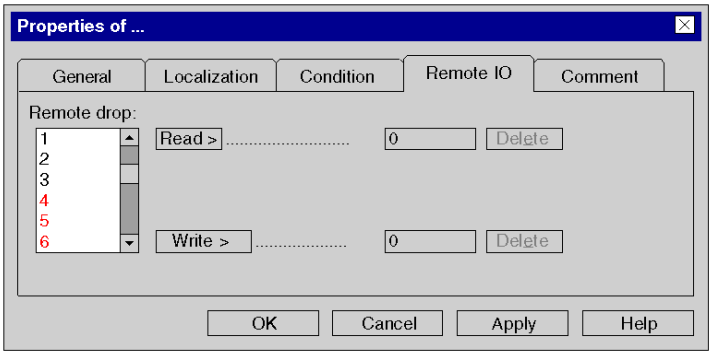



Elements of the **Condition** tab:

| Element | Description |
|---|---|
| **Variable** | You can enter Boolean variables which can conditionally enable/disable the section in this tab.<br>In this area, you can enter the following:<br>● **Name**<br>❍ You can directly enter the name of the variable/address or paste it from the clipboard.<br>❍ Use the command button **...** to open a variable selection dialog box *(see page 330)*.<br>● **Comment**<br>The comments about the selected variables are displayed in this text box. |
| **Can be forced** | This check box indicates whether the selected variable can be forced or not. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections. The generated section will not open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Remote IO tab

Representation of the **Remote IO** tab (only for Quantum projects):

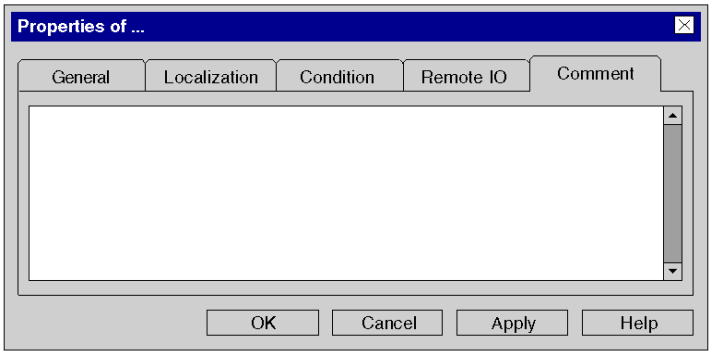



Elements of the **Remote IO** tab:

| Element | Description |
|---|---|
| **Remote drop** | Select the remote drop to read/write from this list box.<br>Already configured drops are displayed in black; non-configured in red.<br>See also section Executing Quantum sections with distributed inputs/outputs *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **Read >** | Use this button to accept the selected drop as the drop to read. I.e. when processing starts on the current section, the inputs for this drop are read. |
| **Write >** | Use this button to accept the selected drop as the drop to write. I.e. when processing ends on the current section, the outputs for this drop are written. |
| **Delete** | Use this command button to reset the selected read/write drops. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections. The generated section will not open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | A comment about the section can be entered in this text box. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections. The generated section will not open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

# Creating SFC sections

## Introduction

SFC sections can be created:

- only in the MAST task
- in offline and online mode

When creating an SFC section, the following three directories are created in addition to the actual SFC section:

- **Unused macros**
  In this directory, all macro sections are stored which were created but are no longer being used *(see page 874)*.
- **Actions**
  This directory contains all Action sections *(see page 944)* for the current SFC section.
- **Transitions**
  This directory contains all transition sections *(see page 957)* for the current SFC section.

## Creating SFC sections

Creating SFC sections:

| Step | Action |
|---|---|
| 1 | Left-click the **Sections** directory for the MAST task or place the cursor on this directory using the arrow keys. |
| 2 | Select (right-click or **Shift**+**F10** key combination) the **New Section** menu command from the shortcut menu.<br>**Result:** The properties dialog box *(see page 470)* for the section is opened. |
| 3 | Enter the section name.<br>The section name must be unique in the entire project and conform to the general naming conventions. |
| 4 | Select the programming language SFC. |
| 5 | If necessary, assign an automation area from 0-15 to the section in the **Area number** list box.<br>This automation area (area number) can be used as sorting criteria in the diagnostic viewer *(see page 1545)*.<br>It makes sense to assign the area number corresponding to the functional unit.<br>**Example:**<br>Cutting: No. 1<br>Milling: No. 2<br>Thread cutting: No. 3<br>In the example, the **Area number** must have the value 1, 2 or 3. |
| 6 | If you want an explicit acknowledgment of all diagnostic results for the SFC section (e.g. exceeding the step monitoring times) to be required in the diagnostic viewer *(see page 1545)*, activate the **Operator control** check box. |

| Step | Action |
|---|---|
| 7 | Select the type of access protection if necessary from the **Protection** list box:<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also Project Protection *(see page 186)*. |
| 8 | Select the functional module to which the section is allocated from the **Localization** tab if necessary.<br>See Functional module *(see page 221)*. |
| 9 | Define a Boolean variable in the **Condition** tab with which the section can be activated/deactivated (section active when variable=1) if required.<br>You have the following options to select variables:<br>● You can enter the name of the variable/address directly or copy it from the clipboard.<br>● Use the command button **...**to open a variable selection dialog box *(see page 330)*.<br>Activate the **Can be forced** check box when you want to force the variable. |
| 10 | **Only applies to Quantum projects:**<br>Assign a read RIO drop and a write RIO drop to the section in the **Remote IO** tab if required.<br>See also Executing Quantum sections with distributed inputs/outputs *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 11 | Enter a comment in the **Comment** tab if desired. |
| 12 | If you wish to create additional sections, confirm the entries with **Apply**. In this case, the Property dialog box will remain open and you can generate additional sections.<br>If you do not wish to create additional sections, confirm the entries with **OK**. In this case, the Property dialog box will close and the generated section will open automatically. |

## Displaying or modifying section properties

Displaying or modifying section properties:

| Step | Action |
|---|---|
| 1 | Left-click on the desired section, or place the cursor on this section with the arrow keys. |
| 2 | Select (right-click or **Shift**+**F10** key combination) the **Properties...** menu command from the shortcut menu.<br>**Result:** The properties dialog box *(see page 470)* for the section is opened. |
| 3 | Implement the desired changes. |
| 4 | Confirm all entries with **OK**. |

## Importing/exporting sections

See sections:

- Importing programs *(see page 1735)*
- Exporting programs *(see page 1733)*

# Properties Dialog Box for SFC Sections

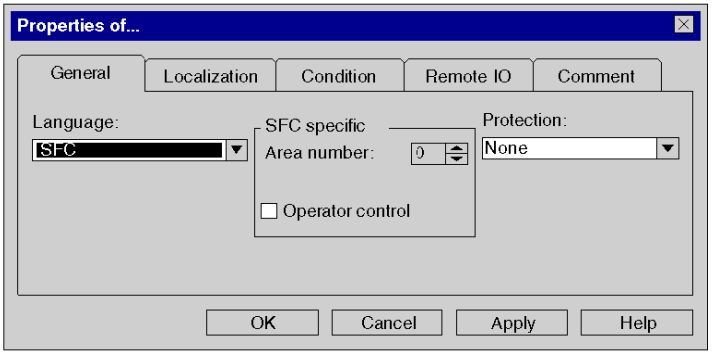
## Structure of the properties dialog box

The properties dialog box consists of five tabs for Quantum projects and four tabs for Modicon M340, Premium, and Atrium projects:

- **General**
  You can enter the name, the programming language, the type of access protection and diagnostic settings for the section in this tab.
- **Localization**
  You can enter the functional module of the section in this tab.
- **Condition**
  You can enter a Boolean variable which can enable/disable the section in this tab.
- **Remote IO**
  This tab is only available for Quantum projects.
  Assign a read RIO drop and a write RIO drop to the section in this tab.
  Assigning drops will remove them from the TASK I/O update routine and will be managed from the section.
  When disabling a section by a condition, the read and write of the assigned drops are still active.
- **Comment**
  You can enter a comment in this tab.

## General tab

Representation of the **General** tab:

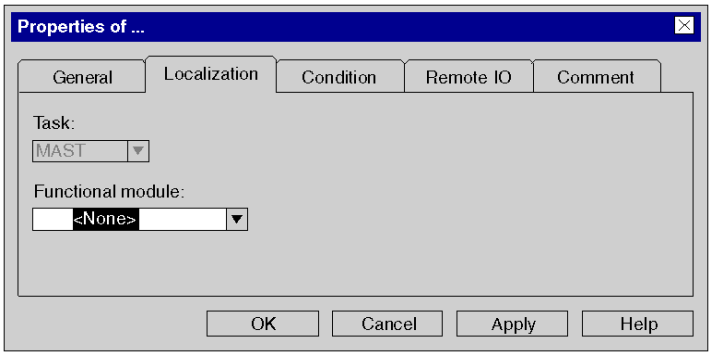


Elements of the **General** tab:

| Element | Description |
|---|---|
| **Name** | Enter the name of the section in this text box.<br>The section name must be unique in the entire project and conform to the general naming conventions. |
| **Language** | Select the programming language of the section from this list box.<br>**Note:** The programming language can only be defined for new sections. The programming language cannot be changed for existing sections. |
| **Area number** | This list box is used to assign an automation range to the section; see also how to create an SFC section *(see page 467)*.<br>This automation area (area number) can be used as sorting criteria in the diagnostic viewer *(see page 1545)*.<br>The valid range for **Area number** 0...15. |
| **Operator control** | When you enable this check box, any diagnostics events occurring in the SFC section (e.g. violation of the step supervision times) must be acknowledged in the diagnostics viewer *(see page 1545)*. |
| **Protection** | Select the type of access protection from this list box.<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also Project Protection *(see page 186)* . |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Localization tab

Representation of the **Localization** tab:

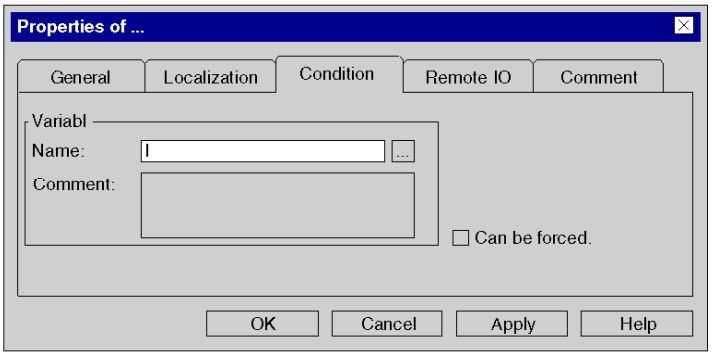



Elements of the **Localization** tab:

| Element | Description |
|---|---|
| **Task** | The task of the section is displayed in this field. |
| **Functional module** | Select the functional module to which the section is to be allocated to in this list box.<br>See also Functional module *(see page 221)*. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Condition tab

Display of the **Condition** tab:



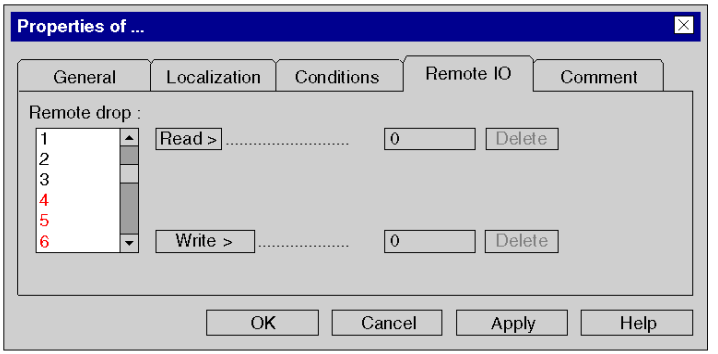

Elements of the **Condition** tab:

| Element | Description |
|---|---|
| **Variable** | You can enter a Boolean variable which can conditionally enable/disable the section in this tab.<br>In this area, you can enter the following:<br>● **Name**<br>❍ You can directly enter the name of the variable/address or paste it from the clipboard.<br>❍ Use the command button **...** to open a variable selection dialog box *(see page 330)*.<br>● **Comment**<br>The comments about the selected variables are displayed in this text box. |
| **Can be forced** | This check box indicates whether the selected variable can be forced or not. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |

| Element | Description |
|---|---|
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Remote IO tab

Representation of the **Remote IO** tab (only for Quantum projects):



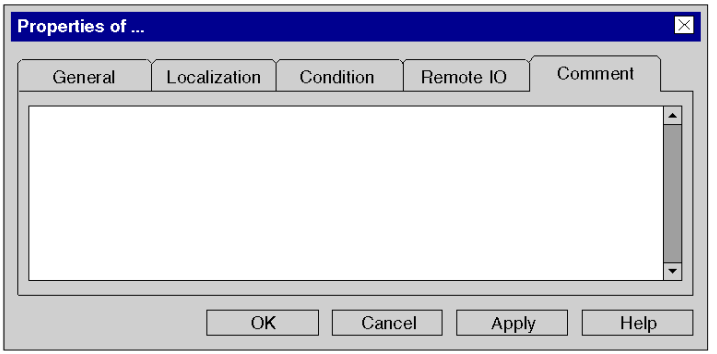

Elements of the **Remote IO** tab:

| Element | Description |
|---|---|
| **Remote drop** | Select the remote drop to read/write from this list box.<br>Already configured drops are displayed in black; non-configured in red.<br>See also section Executing Quantum sections with distributed inputs/outputs *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **Read >** | Use this button to accept the selected drop as the drop to read. I.e. when processing starts on the current section, the inputs for this drop are read. |
| **Write >** | Use this button to accept the selected drop as the drop to write. I.e. when processing ends on the current section, the outputs for this drop are written. |
| **Delete** | Use this command button to reset the selected read/write drops. |

| Element | Description |
|---|---|
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:



Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | A comment about the section can be entered in this text box. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

# Creating action, transition and macro sections

## Creating action, transition and macro sections

The descriptions are found in the following sections:

- Creating action sections *(see page 944)*
- Creating transition sections *(see page 957)*
- Creating macro sections *(see page 916)*

## Displaying or modifying section properties

Displaying or modifying section properties:

| Step | Action |
|---|---|
| 1 | Left-click the target section, or place the cursor on this section with the arrow keys. |
| 2 | Select (right-click or **Shift+F10** key combination) the menu command **Properties** from the shortcut menu.<br>**Result:** The properties dialog box *(see page 478)* for the section is opened. |
| 3 | Modify the section name as required.<br>Note that when you modify the name, the name of a link between the section and the element calling (action, transition or macro section) is deleted. |
| 4 | Select the type of access protection if necessary from the **Protection** list box:<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also section *Section and Subroutine Protection, page 186*. |
| 5 | Enter a comment in the **Comment** tab if desired. |
| 6 | Confirm all entries with **OK**. |

# Action, transition and macro sections properties dialog box

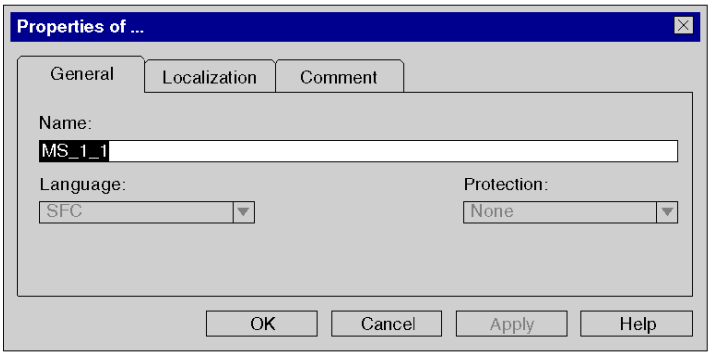
## Structure of the properties dialog box

The properties dialog box consists of three tabs:

- **General**
  The name and the programming language are displayed in this tab. You can also define the type of protection for the section.
- **Localization**
  You can enter the functional module of the section in this tab.
- **Comment**
  You can enter a comment in this tab.

## General tab

Representation of the **General** tab:



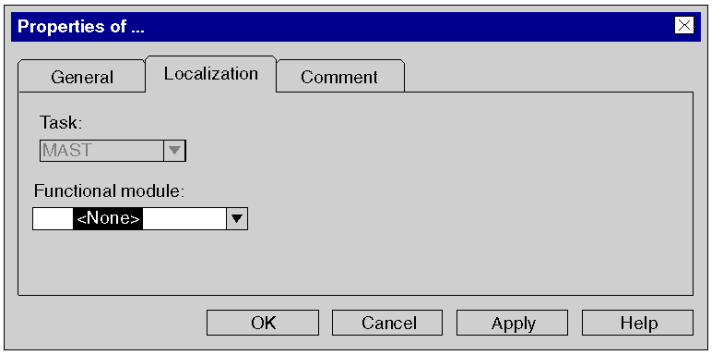

Elements of the **General** tab:

| Element | Description |
|---|---|
| **Name** | The name of the section is displayed in this field.<br>Note that when you modify the name, the name of a link between the section and the element calling (action, transition or macro section) is deleted. |
| **Language** | The programming language of the section is displayed in this list box.<br>**Note:** The programming language can only be defined when creating the section. The programming language cannot be changed for existing sections. |

| Element | Description |
|---|---|
| **Protection** | Select the type of access protection from this list box.<br>• **None**<br>No protection<br>• **Read-Only**<br>Write protection<br>• **No read & write**<br>Read and write protection<br>See also section *Section and Subroutine Protection, page 186*. |
| **OK** | Use this command button to accept all entries and close the properties dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Localization tab

Representation of the **Localization** tab:

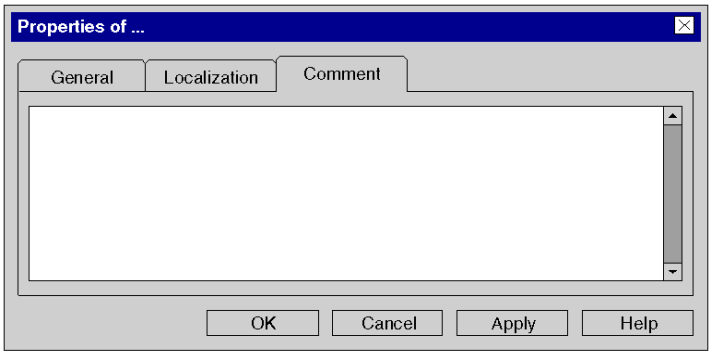



Elements of the **Localization** tab:

| Element | Description |
|---|---|
| **Task** | The task of the section is displayed in this field. |
| **Functional module** | Select the functional module to which the section is to be allocated to in this list box.<br>See also section *Functional modules, page 221*. |
| **OK** | Use this command button to accept all entries and close the properties dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box.<br>**Note:** This command button is only available in the properties dialog box for existing sections. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | A comment about the section can be entered in this text box. |
| **OK** | Use this command button to accept all entries and close the properties dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box.<br>**Note:** This command button is only available in the properties dialog box for existing sections. |

# Create a Sub-Program Section (SR)

## Introduction

Subroutine sections can be created:

- in offline and online mode

Subroutines are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow subroutines** check box.

## Information for creating LD sections

An LD section contains 11-64 columns and 17-2000 lines.

The number of columns is set in the **Tools** → **Project Settings** dialog in the **Editors** tab in the **Number of Columns** text box.

This setting only applies to sections to be created, not existing sections. Always define the number of columns before you create a new LD section.

(The number of lines can be increased or reduced during the creation of section contents.)

## Creating subroutine sections

Creating subroutines:

| Step | Action |
|---|---|
| 1 | Left-click the **SR Sections** directory for the target task, or place the cursor on this directory using the arrow keys. |
| 2 | Select (right-click or **Shift**+**F10** key combination) the **New SR Section** menu command from the shortcut menu.<br>**Result:** The properties dialog box *(see page 483)* for the section is opened. |
| 3 | Enter the section name.<br>The section name must be unique in the entire project and conform to the general naming conventions. |
| 4 | Select the programming language of the section. |
| 5 | Enter a comment in the **Comment** tab if desired. |
| 6 | If you wish to create additional sections, confirm the entries with **Apply**. In this case, the Property dialog box will remain open and you can generate additional sections.<br>If you do not wish to create additional sections, confirm the entries with **OK**. In this case, the Property dialog box will close and the generated section will open automatically. |

## View or edit section properties

Displaying or modifying section properties:

| Step | Action |
|---|---|
| 1 | Left-click on the desired section, or place the cursor on this section with the arrow keys. |
| 2 | Click the right mouse button or press **Shift**+**F10** to select the **Properties menu command in the context menu...**.<br>**Result:** The properties dialog box *(see page 483)* for the section is opened. |
| 3 | Implement the desired changes. |
| 4 | Confirm all entries with **OK**. |

## Importing/exporting sections

See sections:

- *Program Import, page 1735*
- *Exporting the program, page 1733*

# Property Dialog of the Sub-Program Sections

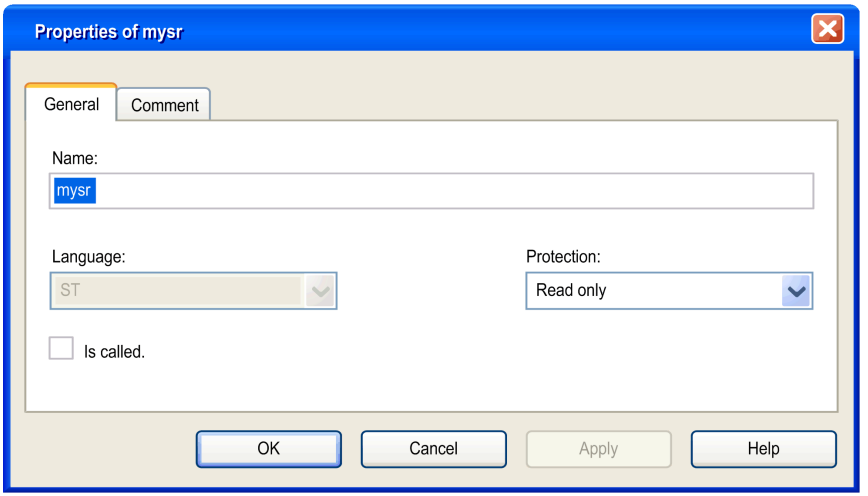
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **General**
  Enter the name and the programming language of the section in this tab. The display indicates whether the subroutine is used in the project.
- **Comment**
  You can enter a comment in this tab.

## General tab

Representation of the **General** tab:



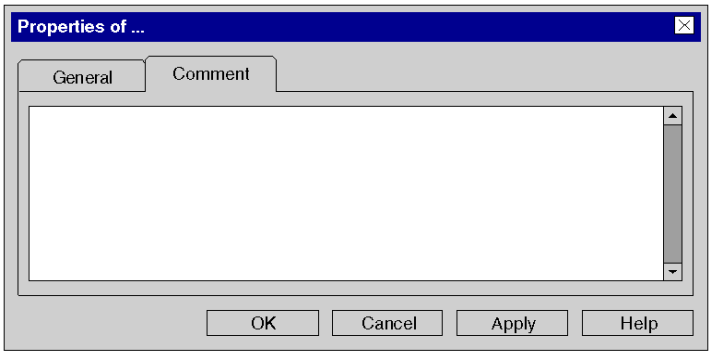

Elements of the **General** tab:

| Element | Description |
|---|---|
| **Name** | Enter the name of the subroutine section in this text box.<br>The section name must be unique in the entire project and conform to the general naming conventions. |
| **Language** | The programming language of the section is displayed in this list box.<br>**Note:** The programming language can only be defined when creating the section. The programming language cannot be changed for existing sections. |

| Element | Description |
|---|---|
| **Protection** | Select the type of access protection from this list box.<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also section *Section and Subroutine Protection, page 186*. |
| **Is called** | This check box indicates whether the subroutine is used in the project. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment**tab:



Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | A comment about the section can be entered in this text box. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

# Creating a Timer Event Section

## Introduction

Timer event sections can only be created in offline mode.

The following can be modified in offline mode:
- the number of the timer event,
- the time base,
- the unit of the preset value,
- the unit for the phase,
- the type of access protection,
- the functional module,
- the comment.

In online mode in stop mode, the following can be changed:
- the unit of the preset value,
- the unit for the phase,
- the type of access protection,
- the functional module,
- the comment.

## Information for creating LD sections

An LD section contains 11-64 columns and 17-2000 rows.

The number of columns is set in the **Tools** → **Project Settings** dialog in the **Editors** tab in the **Ladder Daigram (LD)** area at **Number of columns**.

This setting only applies to sections to be created, not existing sections. Always define the number of columns before you create a new LD section.

(The number of rows and columns can be changed during the creation of section contents.)

## Information for creating FBD sections

An FBD section contains 86,400 grids (number of rows x number of columns).

The number of rows and columns is set in the **Tools** → **Project Settings** dialog in the **Editors** tab in the **Function Block Diagram (FBD)** area.

This setting only applies to sections to be created, not existing sections. Always define the number of rows and columns before you create a new FBD section.

(The number of rows and columns can be changed during the creation of section contents.)

## Creating timer event sections

Creating timer event sections:

| Step | Action |
|---|---|
| 1 | Left-click the **Timer Events** directory of the **Events** task, or place the cursor on this directory with the arrow keys. |
| 2 | Select (right-click or **Shift**+**F10** key combination) the **New Event Section** menu command from the shortcut menu.<br>**Result:** The properties dialog box *(see page 489)* for the section is opened. |
| 3 | Enter the number of the timer event in the **Timer number** list box.<br>The maximum number of timer events depends on the selected CPU, see also section Timer section *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 4 | Enter the time base for the preset value and phase in the **Time base** list box.<br>**Note:** Select the largest time base to reduce system load. |
| 5 | Enter the units for the preset value in the **Preset** text box.<br>The valid range is 0...1023.<br>Preset value = **Preset** x **Timer base** |
| 6 | Enter the unit for the phase in the **Phase** text box.<br>The valid range is 0...1023.<br>Phase value = **Phase** x **Timer base** |
| 7 | Select the programming language of the section. |
| 8 | Select the type of access protection if necessary from the **Protection** list box:<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also section *Section and Subroutine Protection, page 186*. |
| 9 | Select the functional module to which the section is allocated from the **Localization** tab if necessary.<br>See also section *Functional modules, page 221*. |
| 10 | Enter a comment in the **Comment** tab if desired. |
| 11 | If you wish to create additional sections, confirm the entries with **Apply**. In this case, the Property dialog box will remain open and you can generate additional sections.<br>If you do not wish to create additional sections, confirm the entries with **OK**. In this case, the Property dialog box will close and the generated section will open automatically. |

## Displaying or modifying the section properties

Displaying or modifying section properties:

| Step | Action |
|---|---|
| 1 | Left-click on the desired section, or place the cursor on this section with the arrow keys. |
| 2 | Click the right mouse button or press **Shift**+**F10** to select the **Properties menu command in the context menu...**.<br>**Result:** The properties dialog box *(see page 489)* for the section is opened. |
| 3 | Implement the desired changes *(see page 486)*. |
| 4 | Confirm all entries with **OK**. |

## Importing/exporting sections

See sections:

- *Program Import, page 1735*
- *Exporting the program, page 1733*

# Properties dialog box for Timer Event Sections

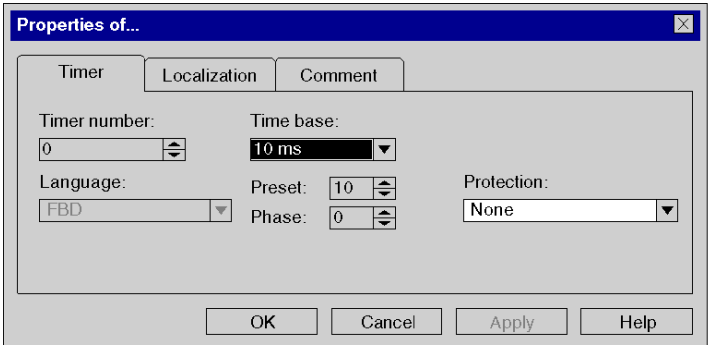
## Structure of the properties dialog box

The properties dialog box consists of three tabs:

- **General**
  In this tab, you can enter the timer event number, the timer time value, the programming language and the type of access protection for the timer event section, see also section Event processing *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
- **Localization**
  You can enter the functional module of the section in this tab.
- **Comment**
  You can enter a comment in this tab.

## General tab

Representation of the **General** tab:



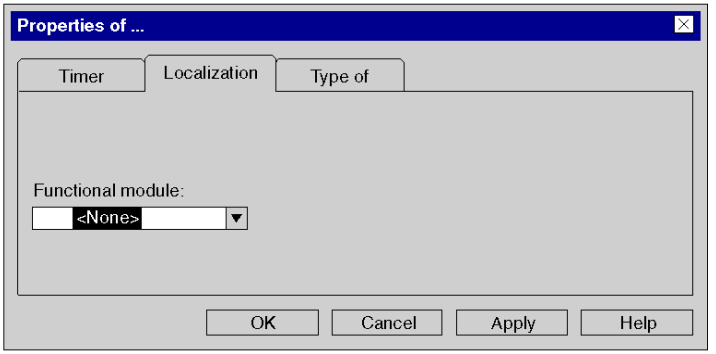
Elements of the **General** tab:

| Element | Description |
|---|---|
| **Timer number** | Enter the number of the timer event in this list box. Note that the ITCNTRL function block *(see Unity Pro, System, Block Library)* is required to activate a timer event.<br>The maximum number of timer events depends on the selected CPU, see also section Event processing *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **Time base** | Enter the time base for the preset value and phase in this list box.<br>The following values are available:<br>● 1 ms<br>● 10 ms<br>● 100 ms<br>● 1 sec<br>The default value is 10 ms.<br>**Note:** Select the largest time base to reduce system load. |
| **Preset** | Enter the unit for the preset value in this text box.<br>The valid range is 1...255.<br>Preset value = **Preset** x **Timer base** |
| **Phase** | Enter the units for the phase in this text box.<br>The valid range is 0...255.<br>Phase value = **Phase** x **Timer base** |
| **Language** | The programming language of the section is displayed in this list box.<br>**Note:** The programming language can only be defined when creating the section. The programming language cannot be changed for existing sections. |
| **Protection** | Select the type of access protection from this list box.<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also Project Protection *(see page 186)*. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |

| Element | Description |
|---|---|
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Localization tab

Representation of the **Localization** tab:

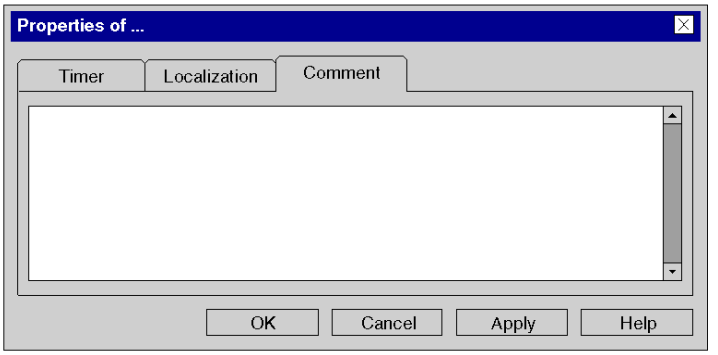



Elements of the **Localization** tab:

| Element | Description |
|---|---|
| **Functional module** | Select the functional module to which the section is to be allocated to in this list box.<br>See also Functional module *(see page 221)*. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | A comment about the section can be entered in this text box. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the properties dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

# Create an I/O Event Section

## Introduction

I/O event sections can be created in offline and online mode.

## Information for creating LD sections

An LD section contains 11-64 columns and 17-2000 rows.

The number of columns is set in the **Tools** → **Project Settings** dialog in the **Editors** tab in the **Ladder Daigram (LD)** area at **Number of columns**.

This setting only applies to sections to be created, not existing sections. Always define the number of columns before you create a new LD section.

(The number of rows and columns can be changed during the creation of section contents.)

## Information for creating FBD sections

An FBD section contains 86,400 grids (number of rows x number of columns).

The number of rows and columns is set in the **Tools** → **Project Settings** dialog in the **Editors** tab in the **Function Block Diagram (FBD)** area.

This setting only applies to sections to be created, not existing sections. Always define the number of rows and columns before you create a new FBD section.

(The number of rows and columns can be changed during the creation of section contents.)

## Creating I/O event sections

Creating I/O event sections:

| Step | Action |
|---|---|
| 1 | Left-click on the **I/O Events** directory of the **Events** task, or place the cursor arrow on this directory. |
| 2 | Use the right mouse button or the **Shift**+**F10** key combination to select the **New Event Section** menu command in the context menu.<br>**Result:** The properties dialog box *(see page 496)* for the section is opened. |
| 3 | Enter the number of the I/O event in the **I/O Event number** list box.<br>The maximum number of I/O events depends on the selected CPU, see also section Event section *(see page 493)* in the Reference manual. |
| 4 | Select the programming language of the section. |
| 5 | Select the type of access protection if necessary from the **Protection** list box:<br>• **None**<br>No protection<br>• **Read-Only**<br>Write protection<br>• **No read & write**<br>Read and write protection<br>See also section *Section and Subroutine Protection, page 186*. |
| 6 | Select the functional module to which the section is allocated from the **Localization** tab if necessary.<br>See also section *Functional modules, page 221* |
| 7 | Enter a comment in the **Comment** tab if desired. |
| 8 | If you wish to create additional sections, confirm the entries with **Apply**. In this case, the Property dialog box will remain open and you can generate additional sections.<br>If you do not wish to create additional sections, confirm the entries with **OK**. In this case, the Property dialog box will close and the generated section will open automatically. |

## Displaying or modifying section properties

Displaying or modifying section properties:

| Step | Action |
|---|---|
| 1 | Left-click on the desired section, or place the cursor on this section with the arrow keys. |
| 2 | Use the right mouse button or press **Shift**+**F10** to select the **Properties** menu command.<br>**Result:** The properties dialog box *(see page 496)* for the section is opened. |
| 3 | Implement the desired changes. |
| 4 | Confirm all entries with **OK**. |

## Importing/exporting sections

See sections:

- *Program Import, page 1735*
- *Exporting the program, page 1733*

# Property Dialog of I/O Event Sections

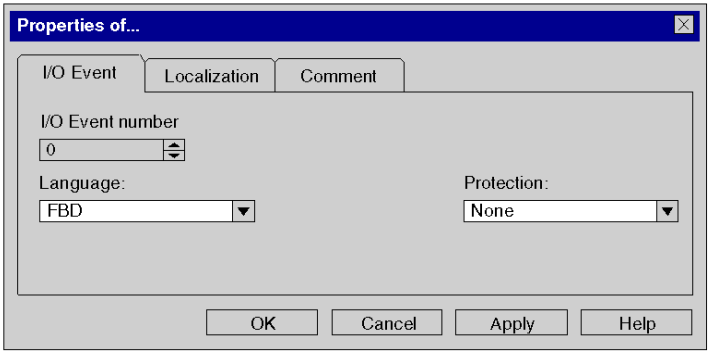
## Structure of the properties dialog box

The properties dialog box consists of three tabs:

- **General**
  Enter the I/O event number, the programming language and the type of access protection for the I/O event section in this tab, see also section Event processing *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
- **Localization**
  You can enter the functional module of the section in this tab.
- **Comment**
  You can enter a comment in this tab.

## General tab

Representation of the **General** tab:



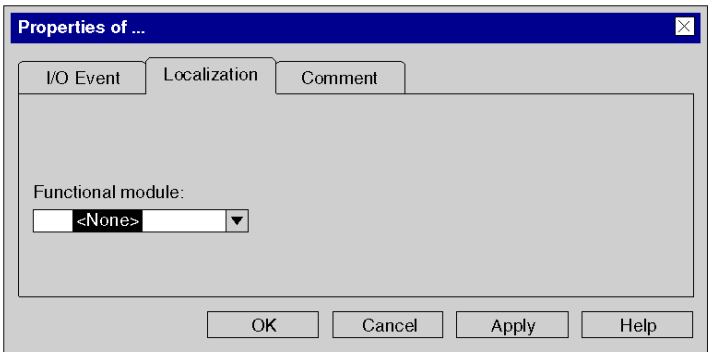

Elements of the **General** tab:

| Element | Description |
|---|---|
| **I/O Event number** | You can enter the number of the I/O event in this list box.<br>The maximum number of I/O events depends on the selected CPU, see also section Event section *(see page 493)* in the Reference manual. |
| **Language** | The programming language of the section is displayed in this list box.<br>**Note:** The programming language can only be defined when creating the section. The programming language cannot be changed for existing sections. |

| Element | Description |
|---|---|
| **Protection** | Select the type of access protection from this list box.<br>● **None**<br>No protection<br>● **Read-Only**<br>Write protection<br>● **No read & write**<br>Read and write protection<br>See also section *Section and Subroutine Protection, page 186*. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Localization tab

Representation of the **Localization** tab:

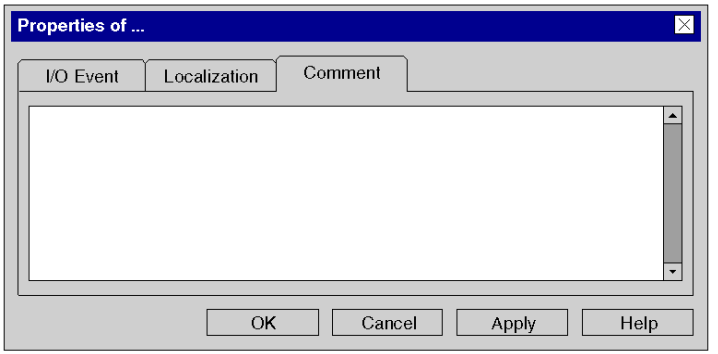


Elements of the **Localization** tab:

| Element | Description |
|---|---|
| **Functional module** | Select the functional module to which the section is to be allocated to in this list box.<br>See also section *Functional modules, page 221*. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment**tab:



Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | A comment about the section can be entered in this text box. |
| **OK** | **Generating a new section:**<br>If you do not want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will close and the generated section will open automatically.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries and close the properties dialog box. |
| **Apply** | **Generating a new section:**<br>If you want to create any additional sections, click on this command button to confirm the entries. In this case, the Property dialog box will remain open and you can generate additional sections.<br>**Displaying properties of an existing section:**<br>Use this command button to accept the entries without closing the properties dialog box. |

# Opening a Section

## Opening a section

Opening a section:

| Step | Action |
|---|---|
| 1 | Click the left mouse button on the section desired (section, subroutine, event), or place the cursor on this section with the arrow keys. |
| 2 | Select (right-click or **Shift**+**F10** key combination) the menu command **Open** from the shortcut menu.<br>or<br>Double click on the section.<br>**Result:** The section is opened. |

# Deleting sections

## Deleting sections

Deleting sections:

| Step | Action |
|---|---|
| 1 | Left-click the target section (section, subroutine, event), or place the cursor on this section with the arrow keys. |
| 2 | Select (right-click or **Shift**+**F10** key combination) the **Delete** menu command from the shortcut menu.<br>**Result:** The section is deleted. |

# Execution order of sections

## Introduction

The execution of sections is carried out according to the order displayed in the structural view.

The arrangement of the actions, transition, macro steps, subroutines, and event sections has no influence on the execution order as their execution is program or event controlled.

The execution order of sections can be changed exclusively in the structural view.

The changes can be made:
- in offline mode
- in online mode in stop mode

## Modifying the execution order

Modifying the execution order:

| Step | Action |
|---|---|
| 1 | Click on the desired section with the left mouse button and hold the mouse button down. |
| 2 | Move the section to the required position within the current task or into another task. |

# Creating LL984 Segments and Networks

## Introduction

The logic of LL984 is programmed in networks.

Several networks are grouped in segments.

Each network contains a matrix of 7 rows and 11 columns, where you can place contacts, coils, function blocks etc.

At the beginning of each segment's code execution the inputs of a RIO drop can be read and at the end the outputs of the same or another RIO drop can be written.

## Recommendations

It is recommended not to use more than 64 segments and 5,000 networks in an application.

## Restrictions

For creating LL984 segments and networks the following rules apply:

- LL984 segments can only be created when the **Ladder Logic 984 (LL984)** project setting is activated via **Tools** → **Project Settings** → **Program** → **Languages**.
- LL984 segments can only be created in the **MAST** task of the application, they are not allowed in the **FAST** and **AUX** tasks.
- Only one LL984 segment can be created inside the subroutine (**SR Sections**) node of the **MAST** task.
- LL984 segments can not be created under **Timer Events** or **I/O Events** nodes.
- LL984 segments can not be created in DFBs.
- For action and transition sections in SFC, LL984 is not allowed as language.
  Only ST, IL, FBD and LD are allowed as language.
- LL984 segments and sections with Unity Pro IEC languages can be mixed inside the **Sections** node of the **MAST** task. The same is valid inside the subroutine (**SR Sections**) node of the **MAST** task.
- LL984 networks can only be created as children of LL984 segments.
- Only LL984 networks are allowed as children of LL984 segments, no sections with IEC languages can be created there.

## Creating a Segment

| Step | Action |
|---|---|
| 1 | Left-click the **Sections** directory of the **MAST** task, or place the cursor using the arrow keys on this directory. |
| 2 | Click the right mouse button or use **Shift+F10** to select **New Section** in the context menu.<br>**Result:** The properties dialog for the segment is opened. |
| 3 | Enter the section (segment) name.<br>The section (segment) name must be unique in the entire project and conform to the general naming conventions of Unity Pro. |
| 4 | Select **LL984 Segment** as programming **Language** of the segment. |
| 5 | Select the functional module to which the segment is attached from the **Localization** tab if necessary.<br>Also refer to *Functional modules* *(see page 221)*. |
| 6 | Define a boolean variable in the **Condition** tab with which the segment can be activated/deactivated (segment active when variable=1) if required.<br>You have the following options to select variables:<br>● Enter the name of the variable/address directly or copy it from the clipboard.<br>● Use the **...** button to open a **Data Selection** dialog. |
| 7 | **Only applies to Quantum projects:**<br>Assign a read RIO drop and a write RIO drop to the segment in the **Remote IO** tab if required. |
| 8 | Enter a comment in the **Comment** tab if desired (max. 1024 characters). |
| 9 | If you wish to create additional segments, confirm the entries with **Apply**. In this case, the dialog will remain open and you can generate additional segments.<br>If you do not wish to create additional segments, confirm the entries with **OK**. In this case, the dialog will close and the generated segment is displayed as the last element in the **Sections** directory.<br>**NOTE:** The position of sections and segments inside the **Structural view** matches exactly the execution sequence and can be changed via drag-and-drop. |

## Creating a Network

| Step | Action |
|---|---|
| 1 | Left-click an **LL984 Segment** in the **Sections** directory of the **MAST** task, or place the cursor using the arrow keys on this directory. |
| 2 | Click the right mouse button or use **Shift**+**F10** to select **New 984 Network** in the context menu.<br>**Result:** The properties dialog for the network is opened. |
| 3 | Enter the network name.<br>The network name must be unique in the entire project and conform to the general naming conventions of Unity Pro. |
| 4 | **LL984 Network** is preselected for the **Language** of the network and can not be changed. |
| 5 | Define a boolean variable in the **Condition** tab with which the segment can be activated/deactivated (segment active when variable=1) if required.<br>You have the following options to select variables:<br>● Enter the name of the variable/address directly or copy it from the clipboard.<br>● Use the command button **...**to open a **Data Selection** dialog. |
| 6 | Enter a comment in the **Comment** tab if desired. |
| 7 | Confirm the entries with **OK**. The dialog will close and the generated network is displayed as the last element in the **LL984 Segment** directory.<br>**NOTE:** The position of networks inside the **Structural view** matches exactly the execution sequence and can be changed via drag-and-drop. |

A new network is normally inserted as the last element in the **LL984 Segment** directory.

To insert a new network directly before a selected network use **Insert Network …** from the context menu.

## Creating a Network for Subroutines

| Step | Action |
|---|---|
| 1 | Left-click the **SR Sections** directory of the **MAST** task, or place the cursor using the arrow keys on this directory. |
| 2 | Click the right mouse button or use **Shift**+**F10** to select **New SR Section** in the context menu.<br>**Result:** The properties dialog for the SR section (**LL984 Segment**) is opened. |
| 3 | Enter the SR section (LL984 segment) name.<br>The SR section (**LL984 Segment**) name must be unique in the entire project and conform to the general naming conventions of Unity Pro. |
| 4 | Select **LL984 Segment** as programming **Language** of the segment.<br>**NOTE:** There can be only 1 LL984 SR segment in a project. So when there is already 1 existing you can no longer select **LL984 Segment** as programming **Language** for a new SR section. |
| 5 | Enter a comment in the **Comment** tab if desired (max. 1024 characters). |
| 6 | Confirm the entries with **OK**. The dialog will close and the generated LL984 segment is displayed in the **SR Section** directory. |
| 7 | Left-click the **LL984 Segment** in the **SR Sections** directory of the **MAST** task, or place the cursor using the arrow keys on this directory. |
| 8 | Click the right mouse button or use **Shift**+**F10** to select **New 984 Network** in the context menu.<br>**Result:** The properties dialog for the network is opened. |
| 9 | Enter the network name.<br>The network name must be unique in the entire project and conform to the general naming conventions of Unity Pro. |
| 10 | **LL984 Network** is preselected for the **Language** of the network and can not be changed. |
| 11 | Enter a comment in the **Comment** tab if desired. |
| 12 | Confirm the entries with **OK**. Tthe dialog will close and the generated network is displayed as the last element in the **LL984 Segment** directory.<br>**NOTE:** The position of networks inside the **Structural view** matches exactly the execution sequence and can be changed via drag-and-drop. |

## Displaying or Modifying Segment/Network Properties

| Step | Action |
|---|---|
| 1 | Left-click on the desired segment/network, or place the cursor on this section/network with the arrow keys. |
| 2 | Click the right mouse button or use **Shift**+**F10** to select **Properties** in the context menu.<br>**Result:** The **Properties** dialog for is opened. |
| 3 | Implement the desired changes. |
| 4 | Confirm all entries with **OK**. |

## Exporting/Importing Segments/Networks

Please refer to Exporting/Importing LL984 Segments/Networks *(see page 1759)*.

# Modifying LL984 Segment Properties

## Introduction

This section describes the specifics of the LL984 Editor referring to modifying LL984 segment properties.

For standard features please refer to *Accessing Sections* *(see page 457)*.

## Protection

The **Protection** property is not available at the moment.

## Localization

You can only attach LL984 segments to a functional module and **not** LL984 networks. The networks of a segment are implicitly attached to the functional module of their segment.

You can attach an LL984 segment to a functional module either in the **Localization** tab of its properties dialog or in the **Project Browser** via drag-and-drop from the **Structural View** to the **Functional View**.

You can detach an LL984 segment from a functional module either in the **Localization** tab of its properties dialog or by selecting **Detach** from the segment´s context menu.

## Condition

You can modify the **Condition** of an LL984 segment in the **Condition** tab of its properties dialog.

When you configure a boolean variable in this tab, the state of this variable is checked at the beginning of the LL984 segment's code execution.

When the variable´s state is 0, the segment and all its networks are not executed.

When the variable´s state is 1 or when no condition variable is defined for the segment, it depends on the condition variables of the networks whether the networks are executed or not.

## Remote I/O

You can synchronize the Remote I/Os of an LL984 segment in the **Remote I/O** tab of its properties dialog.

Here you can configure which Remote Quantum drop is read before and which one is written after execution of the LL984 segment.

Drops which are not yet configured are displayed in red inside the list box.

Not configured drops can be selected here, but when they are not configured before next **Build**, the build will not be successful and a diagnostic message will be displayed.

**NOTE:** This property is only available for Quantum projects.

### Comment

You can enter a comment to an LL984 segment in the **Comment** tab of its properties dialog.

The maximum size of a comment is 1024 characters.

### The LL984 Subroutine Segment

You can only have 1 LL984 segment under the **SR sections** node of the **MAST** task.

The properties dialog of this LL984 subroutine segment only provides the **General** and the **Comments** tabs.

The **Is called** check book in the **General** tab is greyed and not checked.

This is done because in LL984, subroutines can be called via a memory word and so you only know at run time whether a subroutine is called or not.

You can not define a condition variable for the LL984 subroutine segment.

# Modifying LL984 Network Properties

## Introduction

This section describes the specifics of the LL984 Editor referring to modifying LL984 network properties.

For standard features please refer to *Accessing Sections* *(see page 457)*.

## Protection

The **Protection** property is not available at the moment.

## Localization

You can only attach an LL984 segment to a functional modules. The networks of a segment are implicitly attached to the functional module of their segment.

## Condition

You can modify the **Condition** of an LL984 network in the **Condition** tab of its properties dialog.

When you configure a boolean variable in this tab, the state of this variable is checked at the beginning of the LL984 network's code execution.

When the variable´s state is 0, the variable and its networks are not executed.

When the variable´s state is 1 or when no condition variable is defined for the network, it depends on the condition variables of the networks whether the networks are executed or not.

You can define a condition variable for each network.

**NOTE:** Condition variables do not have influence on the execution of an `L9_SKP` block, that means when you have a skip over 4 networks, you skip the remainder of the current network and the next 3 networks independent of their condition variables.

## Remote I/O

You can synchronize the Remote I/Os only for LL984 segments and not for LL984 networks.

## Comment

You can enter a comment to an LL984 network in the **Comment** tab of its properties dialog.

The maximum size of a comment is 1024 characters.

### User Directory/Hyperlink

For LL984 networks you can add hyperlinks by selecting **Add Hyperlink** from the network´s context menu.

For details please refer to chapter *Hyperlinks* *(see page 1817)*.

You can not add user directories to LL984 networks.

### LL984 Networks in Subroutine Segments

The properties dialog of an LL984 network in a subroutine (SR) segment only provides the **General** and the **Comments** tabs.

The **Is called** check box in the **General** tab is greyed and not checked, because in LL984 subroutines can be called via a memory word and so you only know at run time whether a subroutine is called or not.

You can not define a condition variable for an LL984 network in a subroutine (SR) segment.

# Scheduling LL984 Segments and Network

## Introduction

Sections and LL984 segments (and also LL984 networks inside segments) are executed in the sequence in which they are displayed in the **Structural View** of the **Project Browser**.

To change this sequence, you can drag-and-drop sections, LL984 segments or LL984 networks inside the **Structural View**.

## Restrictions

For dragging-and-dropping LL984 segments and networks the following rules apply:

- Sections (ST, LD, …) can not be dropped into LL984 segments.
- LL984 segments can not be dropped into other LL984 segments.
- LL984 networks can only be dropped into LL984 segments.
  LL984 networks can be dropped into the LL984 subroutine segment under the **SR Sections**) node of the **MAST** task.
  After dropping the networks, they are re-analyzed latest during next **Build**, because e.g. `L9_LAB` function blocks are only allowed in networks of the LL984 segment in the **SR Sections** node.
- When an LL984 network has an activation condition and is dragged-and-dropped into the LL984 subroutine segment, the LL984 network loses its activation condition because subroutine networks do not have activation conditions.

## Drag-and Drop Between Applications

You can not drag-and drop LL984 segments and networks from one Unity Pro application to another.

To copy sections between applications you have to use the import/export feature of Unity Pro.

# Deleting, Cutting, Copying and Pasting LL984 Segments and Network

## Deleting LL984 Segments and Networks

You can delete LL984 segments and networks in the **Structural View** of the **Project Browser**.

To delete an LL984 segment or network either select **Delete** from its context menu or press the **DEL** key.

If you want to delete an LL984 segment, a message is displayed that all its networks will be deleted, too.

You have to confirm this message to delete the LL984 segment.

## Cutting, Copying and Pasting

You can not cut, copy and paste LL984 segments and networks in a Unity Pro application.

Alternatively you can use the import/export feature on segment or network level.

# Chapter 19
# Unity Pro Software Options

## Subject of this Chapter

This chapter describes the options in Unity Pro software. The software provides 3 types of option:

- **Project Settings**: they are specific to the generated project and to how it is executed on the PLC (for example: acknowledgment of diagnostic functions, execution of SFC language, etc.).
- **Options..**: they are specific to the workstation and may therefore differ from one station to another for the same project (for example: the way in which errors are displayed, direction of movement of selection after data entry, etc.)
- **Custom..**: they are used to customize the ergonomics of the workstation (contents of toolbar etc.).

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 19.1 | Project Settings | 516 |
| 19.2 | Options | 554 |
| 19.3 | Customize Dialog Box | 571 |

# Section 19.1
# Project Settings

## Overview

This section describes the **Tools** → **Project settings** dialog box tab.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| General Description | 517 |
| General Project Settings | 519 |
| Variables | 531 |
| Program | 534 |
| Maintain Output Links on Disabled EF (EN=0) | 542 |
| Configuration Settings for Operator Screens | 550 |

# General Description

## Introduction

| ⚠ CAUTION |
| --- |
| **UNEXPECTED APPLICATION BEHAVIOR**<br>When importing a project settings file, ensure that the required file is used. Importing an inappropriate project setting file can result in unexpected application behavior.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

The table below describes the general functions used with all Project Settings tabs:

| Fields | Description |
| --- | --- |
| **Import** | This button allows the selection of a Project Setting .XSO file. All values in the file will be taken into account for the current opened project. After Import, click on OK or Apply to validate the settings. |
| **Export** | This button allows the current values of the project settings to be saved in an .XSO file. |
| **Reset All** | This button is used to reset all the settings to their default values. |
| **OK** | This button is used to save the modifications and close the current tab. |
| **Apply** | This button is used to save the modifications. |
| **Cancel** | This button is used to cancel without saving the modifications on the current tab. |
| **Help** | This button is used to call the online help. |

## General Navigation

The table below describes the general key shortcut navigation used with all Project Settings tabs:

| Fields | Description |
| --- | --- |
| **TAB** | This key is used to move from one control to another. |
| **ENTER** | This key is used to confirm the current action. |
| **ESC** | This key is used to cancel without saving the modifications in the current tab. |
| **Up, Down, Left and Right Arrows** | These keys are used to navigate within the tree control and the right list box. |

## Specific Navigation

The table below describes the Specific key shortcut navigation used with all Project Settings tabs:

| Fields | Description |
|---|---|
| **Space** | This key is used to check/uncheck an element. |
| **F2** | This key is used to activate the current line. |
| **ESC** | This key is used to deactivate the current line. |
| **Up and Down Arrows** | These keys are used to navigate in the list box in the active line. |
| **+ and -** | These keys are used to increase or decrease a value in the number field in the active line. |

# General Project Settings

## At a Glance

The following tables describe the available functions in the **Project Settings** dialog for project generation:

- **Management of build messages**
- **Build settings**
- **Project autosaving on download**
- **PLC embedded data**
- **PLC diagnostics**
- **PLC behaviour**
- **Path**
- **Time**
- **Configuration**

## Management of build messages

The table below shows all available **Management of build messages** fields, their description and required action to take modification into account:

| Fields | Description | |
|---|---|---|
| **Variable not used generates** | **nothing** or a detected **warning** is displayed in the view window if variables defined in the project are not used in the program.<br>Variables defined in the project but are not used in the program. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **Multiple writing of variables generates** | Variables assigned several times to different places in the program. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **Parameters not assigned generates** | EFB or DFB parameters not assigned. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Multiple use of FB instance generates** | EFB or DFB instances used several times at different places in the program. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |

| Fields | Description | |
|---|---|---|
| **DDT mapping compatibility generates** | An alignment test on DDTs is done for each type of data structure (BOOL, BYTE, INT, DINT...) with specific rules for each platform. Unity Pro displays a notification statomg which DDTs have to be modified when either an **Analyze Project**, **Build Changes**, or **Rebuild all Project** command runs.<br>If the data structures have the same alignment in each platform, communication between them is possible. Otherwise, a notification appears to inform which DDTs have to be modified for allowing communication.<br>Premium and Quantum have the same mapping rule so these two platforms are aligned for communication.<br>M340, M580 and Momentum has a different mapping rule than Premium and Quantum. So, it is necessary to check if the alignment of the exchanged DDTs is the same for each platform (M340 and Premium for example) and in this case modify the DDTs.<br>Refer to DDT: Mapping rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* for alignment rules. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **Overlapping of address generates** | If **nothing** is selected, no message is displayed in the view window even if there is overlapping of address of some variables.<br>If a detected **warning** is selected, some address messages are displayed in the view window if there is overlapping of addresses of some variables.The project is generated.<br>If a detected **error** is selected, some detected error messages are displayed in the view window if there is overlapping of addresses of some variables. Due to the detected errors, the project is not generated.<br>**NOTE:** Overlapping of topological addresses is not detected. Only flat addresses are detected by Unity Pro.<br>**NOTE:** The overlapping can generate a high number of detected errors for clients using many located variables or LL984 converted code and so increase the build generation time. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Missing gateway IP @ generates** | A detected error is generated on Quantum RIO architecture when a 140 NOC 780 00, 140 NOC 781 00 or 140 CRP 312 00 module is configured without gateway IP address defined (equal to 0.0.0.0). | |
| | Modification: | Online possible |
| | Action required: | No build necessary |

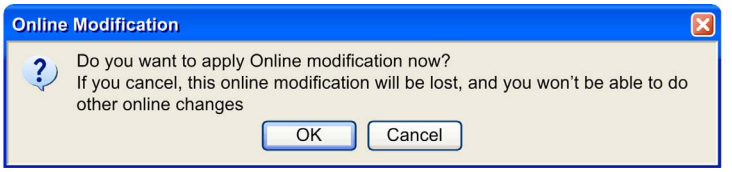
| Fields | Description | |
|---|---|---|
| **Control numbers of warnings** | The **Control numbers of warnings** has to be selected and a value defined in the **Maximum of warnings allowed** field (the valid range is 100...2000). Then, when the number configured is reached, the next detected warnings are not displayed in the view window. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **Control numbers of errors** | The **Control numbers of errors** has to be selected and a value defined in the detected **Maximum of errors allowed** field (the valid range is 100- 2000). Then, when the number configured is reached, the next detected errors are not displayed in the view window. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |

## Build settings

The table below shows all available **Build settings** fields and their description:

| Fields | Description | |
|---|---|---|
| **Create New data block on line** | Exclusive setting that allows you to select the management of data memory during on line modification.<br>This setting allows you to add/modify variables without limitation of the memory block. By default, this check box is checked:<br>● When checked, the dynamic allocation of unlocated variables in a new data block can be done online and you are not prompted Unity Pro does not prompt you to **Rebuild all Project**.<br>● When unchecked, Unity Pro acts as per the free data memory setting, which appears below.<br>**NOTES:**<br>● For applications (STA, XEF, ZEF files, or an upload) that contain this project setting when opened the setting is not changed.<br>● For applications that do not contain this project setting, when opened or uploaded it is set to the default value: checked. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) to deselect<br>**Build Changes** (Online) to select |

| Fields | Description | |
|---|---|---|
| **Free data memory (in kbytes)** | Exclusive setting that allows you to select the management of data memory during on line modification.<br>This setting was the only way to manage data memory during on line changes before Unity Pro V6.0. It is kept for compatibility with previous versions or when only a few modifications of variables on line are planned.<br>When this setting is valid, only one memory block 1...64 K is reserved.<br>Each time unlocated variables are created or modified, they are added to this memory block in the PLC.<br>When the memory allocated is full, it is not possible to perform additional on-line modifications (a detected error message is displayed during the on-line change session). A **Rebuild All Project** is mandatory to restart with this memory block empty and available for new on line modifications. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Virtual connected mode** | Used to modify a project on a non-connected terminal, as if this terminal was connected to the PLC.<br>It is then simply a case of connecting the terminal to the PLC and activating the command **Build** → **Build Changes** in order to have these modifications taken into consideration in the PLC. This transfer does not stop the PLC and only the changes made are taken into account. The purpose of this mode is to inform when an online modification is not possible. However, if the change is made, the virtual connected mode is aborted.<br>Analysis is permitted in this mode, but generation is not possible. The project can be regenerated at any time, but this exits virtual mode. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Generate with LD link animation** | Allows choosing to include or not information on ladder diagram link animation in the generated code. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Optimized code** | Used to reduce the size of the code generated. Note the following:<br>● On Premium PLCs and on Quantum 140 CPU 6•• PLCs, the bit %S18 is no longer managed for arithmetic operations on EDT (Elementary Data Types) and tables.<br>● On Quantum 140 CPU 3••/4••/5•• PLCs, the bit %S18 is no longer managed for arithmetic operations on EDT. The bit %S20 is no longer managed for table indexing (the threshold values are still checked). | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

| Fields | Description | |
|---|---|---|
| **Apply online modification** | Allows choosing whether the application is to be downloaded automatically or on a user request after the **Build Changes** command.<br>On each modification, this dialog box allows you to validate or not the modification:<br>  | |
| | Modification: | Online possible |
| | Action required: | No build necessary |

## Project autosaving on download

The table below shows all available **Project autosaving on download** fields and their description:

| Fields | Description | |
|---|---|---|
| **save STA** | If selected, the archive file is automatically saved each time it is transferred to the PLC (generation in online mode or transfer of complete program). | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **save STU** | If selected, the project is automatically backed up (STU and auto.STA files saved if the application has changed since the last download) each time it is transferred to the PLC (generation in online mode or transfer of complete program).<br>**NOTE:** Each time the project is saved (STU file), if the project is in BUILT state, a file with AUTO.STA suffix is saved, even if the STU box is not checked. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **save XVM** | If selected, the project variables are automatically backed up (XVM file) each time the project is transferred to the PLC (generation in online mode or transfer of complete program). | |
| | Modification: | Online possible |
| | Action required: | No build necessary |

## PLC embedded data

The table below shows all available **PLC embedded data** fields and their description:

| Fields | Description | |
|---|---|---|
| **Data dictionary** | Dictionary of symbolized variables with their characteristics (address, type,...) in the PLC memory.<br>If this setting is enabled, all symbolized variables of the application are embedded in the PLC memory. Any client (SCADA using OFS, HMI, embedded Web server...) can then animate or modify those variables. By default, the settings are disabled.<br>The **Data dictionary** is generated at build time. This block is updated only if it is needed during an online modification.<br>**NOTE:** Comments of the variables are not stored.<br>The 3 following properties improve the performances of the **Data dictionary** online build:<br>● When checking **Preload on build changes**, download of data dictionary reflecting the change of variable definition is performed before applying the real change of the application in the PLC. The PLC client tools are then able to learn new changes of data.<br>● The **Effective Build changes time-out (sec)** is the time (1...240 s) that Unity Pro is waiting between the download of the new data dictionary and the real online application change. The down counting of the time-out is displayed in the progress bar during the **Build Changes** operation.<br>● Checking the **Only HMI variables** and identifying the HMI variables necessary in the **Data Property** table improves online build performances and memory usage of the PLC. This setting means that the data dictionary embeds only variables or DFB/EFB instances that have the **HMI** attribute set.<br>**NOTE:** A detected **Memory full** error message could happen during a **Build Changes** if there is not enough free memory. A project with the **Data dictionary** enabled needs a large amount of free memory to be built. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Online) |
| **Upload information** | If selected, this non-executable code designed to restore the source of the project on a terminal is simultaneously loaded with the executable code when transferring the project to the PLC.<br>If selected, program transfer from the PLC to the PC cannot be done. In this non-executable code, the check boxes **Comments** and **Animation tables** are used to include comments associated with the variables and types and animation tables respectively. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |

| Fields | Description | |
|---|---|---|
| **Upload information management** | If **Automatic** is selected, all online modifications are made on both the executable code and the non-executable code designed to restore the project on the terminal.<br>If **On user request** is selected, the online modifications are only made on the executable code in the PLC processor. The non-executable code has to be updated implicitly using the **PLC** → **Update Upload Information** command.<br>An indicator at the foot of the screen is displayed if the non-executable code has not been updated. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **Optimize data on-line change** | If this box is checked, it reduces the time of the **Upload information** generation relating to variables and function blocks of the application in online mode.<br>**NOTE:** Uncheck this option to keep compatibility with Unity Pro versions before 4.1. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

## PLC diagnostics

The table below shows all available **PLC diagnostics** fields and their description:

| Fields | Description | |
|---|---|---|
| **Application diagnostics** | If the box is checked, you can choose an **Application diagnostic level**. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Application diagnostic level** | Diagnostic level choice:<br>● If **No cause of fault** is selected, the analysis of the diagnostic detected error origin is not performed.<br>● If **Local diagnostic** is selected, the analysis of the diagnostic detected error origin is performed within the section using the diagnostic DFB.<br>● If **Global diagnostic** is selected, the analysis of the diagnostic detected error origin is performed within the whole application. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **System diagnostics** | If the box is checked, you can choose a **System diagnostic language**. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

| Fields | Description | |
|---|---|---|
| **System diagnostic language** | Language choice:<br>● **English** (by default)<br>● **German**<br>● **Italian**<br>● **Spanish**<br>● **French** | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Rack Viewer diagnostics information** | If selected, it provides a view of the rack. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Online) |
| **Rack Viewer variable names** | If selected, it provides a view of the rack names. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Online) |
| **Program Viewer information** | The **Program Viewer** runs on a PC that is connected to an Ethernet module. If selected, it provides a view of the program structure (functional and structural) and displays the contents of sections and animation.<br>Use of this function requires Unity Pro 6.0 of higher.<br>**NOTES:**<br>● Useful only when having a FactoryCast 140 NOE 771 1•, TSY ETY 5103 in the PLC local rack.<br>● The **Data dictionary** has to be selected and preferably without the **Only HMI variables** option.The program viewer relies on the data dictionary being fully present in the PLC to animate the variables in the code. If the data dictionary is not present, or incomplete, the code is visible but not animated or only partially animated. Using the HMI variables only would limit the scope of the animated variables in the code. | |
| | Modification: | Offline only |
| | Action required: | **Build Changes** (Offline) |
| **Event Logging** | Select this field to enables event logging for each possible originator of event logging in the architecture. The syslog settings entered below are transferred to each event logging originator in the architecture.<br>**NOTE:** Event logging is enabled if this field is selected and a valid syslog server IP address is entered in **SYSLOG server address** field below.<br>**NOTE:** Unity Pro event logging is configured in **Security Editor** → **Profiles**. | |
| | Modification: | Offline only |
| | Action required: | **Build Changes** (Offline) |

| Fields | Description | |
|---|---|---|
| **SYSLOG server address** | Sets the IP address of the syslog server that stores the system events logging.<br>**NOTE:** The default value (0.0.0.0) does not allow events to be logged, a valid server address is needed with the **Event Logging** field selected.<br>**NOTE:** The syslog server address entered here is embedded in the application. | |
| | Modification: | Offline only |
| | Action required: | **Build Changes** (Offline) |
| **SYSLOG server port number** | Sets the syslog port number.<br>Port number value:<br>**Default:** 601<br>**Range:** 0...65535 | |
| | Modification: | Offline only |
| | Action required: | **Build Changes** (Offline) |
| **SYSLOG server protocol** | Sets the syslog server protocol.<br>TCP V1 is the default syslog protocol.<br>**NOTE:** | |
| | Modification: | The syslog protocol cannot be modified in Unity Pro 10.0. |
| | Action required: | – |

## PLC behaviour

**⚠ CAUTION**

**UNINTENDED EQUIPMENT OPERATION**

Ensure that the use of the function **Reset %M on Stop->Run transition** is appropriate to the application and carefully tested before any implementation.

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

The table below shows the **PLC behaviour** field and its description:

| Field | Description | |
|---|---|---|
| **Reset %M on Stop->Run transition** | This function resets %M that is not used in LL984 sections during a STOP to RUN transition of the CPU.<br>**NOTE:** This function is only available for the Quantum CPUs with version 2.82 or higher of the operating system. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

The action of the **Reset %M on Stop->Run transition** function depends on the following **(Tools menu) → Project Settings** check boxes:

- **General → PLC behaviour → Reset%M on Stop->Run transition**
  This setting only impacts % M not used in LL984 language sections.
- **Program → Languages → Ladder Logic 984 (LL984)**
  This setting only impacts %M used with normal coils in LL984 language sections.

This table shows that results of a transition depend on which check boxes are validated:

| Reset%M | LL984 | Resulting Action on Stop → Run transition |
|---|---|---|
| Unchecked | Unchecked | No %M are reset. |
| | Checked | Reset of %M used with **normal** LL984 coils. |
| Checked | Unchecked | Reset of %M. |
| | Checked | Reset of all %M not used with **latched** LL984 coils. |

If there are %M conflicts:

- For a %M used on an LL984 **latched** coil **and** on a **non-LL984** coil, the LL984 **latched** coil has the higher priority
- For a %M used on an LL984 **normal** coil **and** on a **non-LL984** coil, the LL984 **normal** coil has the higher priority

If a %M has its value forced, its forced value does not change after a **Reset %M on Stop->Run transition**.

## Path

The table below shows the **Path** field and its description:

| Fields | Description | |
|---|---|---|
| **Hyperlink Root Path** | Allows choosing a path which is used for LL984 application *(see page 118)* when the option to convert long comments to text files is chosen. In this case, the long comments are available as hyperlinks in the variable comments column of the data editor.<br>Additionally you can define a path for the created hyperlink text files by either of 2 ways:<br>● Define a relative path in the root directory of your application.<br>● Use the **...** button to select any other path.<br>The subfolder of the hyperlink files is displayed below the edit field.<br>The path defined is written to the project settings of your new application: **Tools → Project Settings → General → Path → Hyperlink Root Path** | |
| | Modification: | Online possible |
| | Action required: | No build necessary |

## Time

The table below shows all available **Time** fields and their description:

| Fields | Description | |
|---|---|---|
| **Time Stamping Mode** | Defines the time stamping mode:<br>● Applicative *(see Applicative Time Stamping with Unity Pro, User Guide)*<br>● System *(see System Time Stamping, User Guide)* | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Max events stored** | Only available for **System** time stamping mode, this value (between 0 and 4000) defines the maximum number of time stamping events stored in the PLC memory. This maximum number defines the size reserved in the PLC memory for time stamping storage.See the **PLC → Memory Consumption...** to know the memory size used and refine this value if needed.<br>**NOTE:** When building the project, if this value is too small, a detected error message will give the exact value to select. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Custom TimeZone** | Gives the possibility to define your own time zone. If checked the option:<br>● **Time Zone** in UTC time cannot be selected.<br>● **Time Offset** can be adjusted.<br>● **Automatically adjust clock for daylight saving** can be configured. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Time Zone** | Allows you to select the wanted time zone in the list of standard time zone. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Time Offset** | Allows you to add an offset to the **Time Zone** between -1439...1439 minutes. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Automatically adjust clock for daylight saving** | **Enabled**: In the **START Daylight Saving** and **END Daylight Saving** fields, enter the month, day of the week, occurrence range, hour, minute, second, and offset from the respective dropdown lists. **Disabled** (by default): The **START Daylight Saving** and **END Daylight Saving** fields are disabled because their times are automatically changed in the spring and fall every year.<br>**NOTE:** Automatic clock adjust for daylight saving change is not implemented on Modicon Premium range. You are advised not to activate this useless feature (No internal or external action is performed when this function is activated). | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |

**NOTE:** The time settings are used by the BMX CRA 312 •0 and BMX ERT 1604T modules. The Quantum and Premium Ethernet modules have their own time setting configuration screen.

## Configuration

The table below shows the **Configuration** field and its description:

| Fields | Description | |
|---|---|---|
| **M580 preferred I/O data type (Local I/O)** | You can choose the default type of I/O for adding a new device in a Modicon M580 main local rack (containing the CPU):<br>● **Device DDT** (by default): unlocated data<br>● **Topological**: located data | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **M580 preferred I/O data type for Quantum EIO** | You can choose the default type of I/O for adding a new Quantum I/O module in a Modicon M580 remote I/O:<br>● **Device DDT** (by default): unlocated data<br>● **Topological**: located data | |
| | Modification: | Online possible |
| | Action required: | No build necessary |

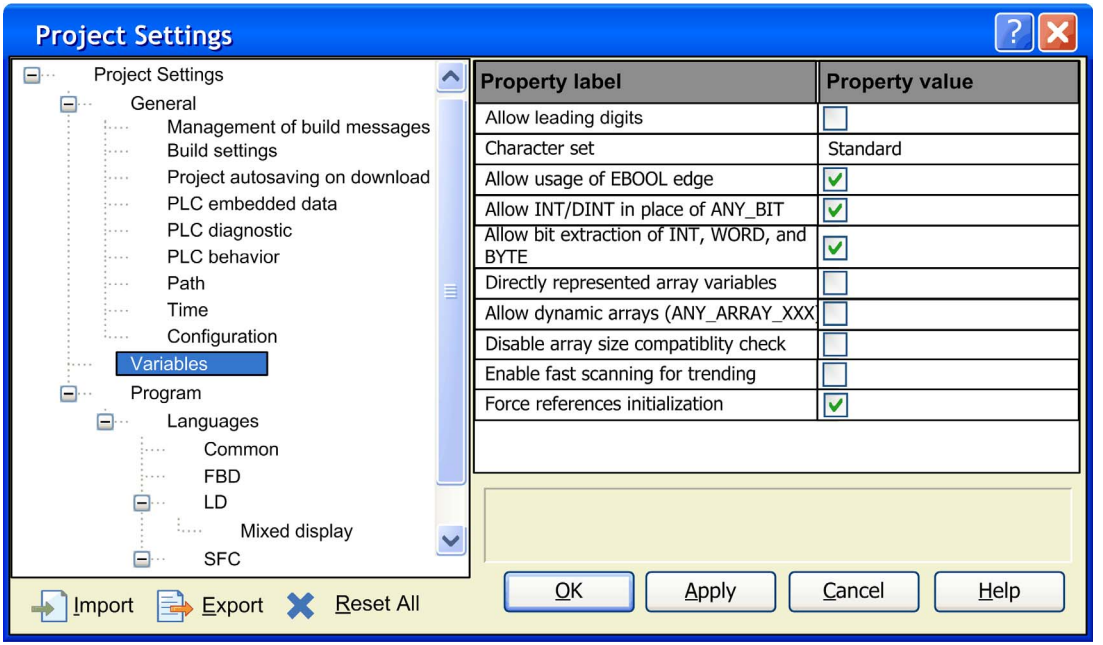
# Variables

## Introduction

This dialog box can be accessed with the **Tools** → **Project Settings** → **Variables**.

## Representation

Representation of the **Variables** section:



Elements of the **Variables** section:

| Element | Description | |
|---|---|---|
| **Allow leading digits** | When activating this check box, leading digits in names can be used (for example, section names, variable names, step names, and so on). Verify that the names contain at least one letter in addition to the leading digits. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Character set** | **Standard**: When activating this optional button, the standard (IEC) character set in names can be used (for example, section names, variable names, step names, and so on). It is not possible to use accented characters (for example 'é').<br>**Extended**: When activating this optional button, the ASCII character set in names can be used (for example, section names, variable names, step names, and so on).It is possible to use accented characters.<br>**Unicode**: When you activate this optional button, the unicode character set in names can be used (for example, section names, variable names, step names, and so on). This option is necessary to create elements in not-roman alphabets like Chinese. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Allow usage of EBOOL edge** | When you activate this check box, the `EBOOL` data type for edge recognition can be used. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow INT/DINT in place of ANY_BIT** | When activating this check box, for **generic** parameters of data type `ANY_BIT`, the parameters of data type `INT` or `DINT` can be used (not `UINT` and `UDINT`).<br>**For example:**<br>**Allowed:**<br>`AND (AnyBitParam := IntVar1, AnyBitParam2 := IntVar2):`<br>**Not allowed:**<br>`AND_WORD (WordParam1 := IntVar1, WordParam2 := IntVar2);` (In this case, use `AND_INT`.<br>`AND_ARRAY_WORD (ArrayInt, ...)` (In this case, perform an explicit type conversion using `INT_ARR_TO_WORD_ARR (...)`. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow bit extraction of INT, WORD and BYTE** | When activating this check box, the individual bits from an `INT`, `WORD`, or `BYTE` can be extracted, see Single bits in words *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

| Element | Description | |
|---|---|---|
| **Directly represented array variables** | When this check box is checked, it is possible to:<br>● Declare references via indexed addresses (for example, `%MW1[3]`).<br>See References using indexed addresses *(see Unity Pro, Program Languages and Structure, Reference Manual )*<br>● Reference bits and words as tables (Array) (for example, %MW1:16).<br>See Table of bits and words *(see Unity Pro, Program Languages and Structure, Reference Manual )*<br>● Use a combination of both of them (for example, %MW1[3]:16). | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow dynamic arrays [ANY_ARRAY_XXX]** | When activating this check box, the dynamic arrays can be used, see Tables *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Disable array size compatibility check** | When activating this check box, no compatibility check is made on the arrays when one array is assigned to another, see also section Tables *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>**Note:** This option applies only to one-dimensional array. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Enable fast scanning for trending** | When activating this check box, a fast scan is made on variables. The analysis of trending variables is improved. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Force references initialization** | When activating this check box (default state), variables of Reference type *(see Unity Pro, Program Languages and Structure, Reference Manual )* have to be initialized when they are created in the Data Editor.<br>When the check box is not activated, if the variable of Reference type are not initialized in the Data Editor, they are initialized by default with the NULL value, then they have to be assigned by the `REF` *(see Unity Pro, Standard, Block Library)* function in the application program.<br>**NOTE:** The application performance is better if the checkbook is activated. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

# Program

## Introduction

This configuration section can be accessed **Tools** → **Project Settings** → **Program**. It contains common languages, FBD, LD, SFC, and ST section display settings.

## Languages

| Element | Description | |
|---|---|---|
| **Function Block Diagram (FBD)** | When this check box is selected, the FBD language can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Ladder (LD)** | When this check box is selected, the LD language can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Sequential Function Chart (SFC)** | When this check box is selected, the SFC language can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **List (IL)** | When this check box is selected, the IL language can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Structured Text (ST)** | When this check box is selected, the ST language can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

| Element | Description | |
|---|---|---|
| **Ladder Logic 984 (LL984)** | When this check box is selected, the LL984 language can be used in your project.<br>The LL984 is not available for Modicon M580 system.<br>**NOTE:**<br>For modifying the Ladder Logic 984 (LL984) setting, the following restrictions apply:<br>● This setting is disabled for applications using Quantum Safety PLCs, as LL984 segments and networks can only be created for non-safety applications.<br>● You have to go offline. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

## Common

| Element | Description | |
|---|---|---|
| **Allow procedures** | When this check box is selected, the procedures can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow subroutines** | When this check box is selected, the subroutines can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow nested comments** | When this check box is selected, the comments can be nested without it being reported as an error. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow multi assignment [a:=b:=c] (ST/LD)** | When this check box is selected, the multi-assignments *(see Unity Pro, Program Languages and Structure, Reference Manual )* can be used in ST. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

| Element | Description | |
|---|---|---|
| **Allow empty parameters in non-formal call (ST/IL)** | When this check box is selected, it is not necessary to assign a value to all formal parameters when making informal calls of functions (ST) and function blocks (IL/ST).<br>See Informal call (ST) *(see Unity Pro, Program Languages and Structure, Reference Manual )* and Informal call (IL) *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Usage of ST expressions (LD/FBD)** | When this check box is selected, the ST expressions can be used as actual parameters on FFB inputs and contacts. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Maintain output links on disabled EF (EN=0)** | When this check box is selected, the behavior of link values can be defined in case an EF is called conditionally (EN input connected). The default value of this option for a new application is "unchecked". After changing this setting a **Rebuild All Project** of the application is necessary, because it influences the generated code.<br>For detailed information, refer to *Maintain Output Links on Disabled EF (see page 542)*.<br>**NOTE:** This setting allows keeping the same behavior as a Concept application. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |
| **Display complete comments of structure element** | Displays the complete comments of structured elements in the mixed mode of LD and in the tooltips for variables in FBD/LD/SFC sections *(see page 597)*. | |
| | Modification: | Offline only |
| | Action required: | **Rebuild All Project** (Offline) |
| **Enable implicit type conversion** | When this check box is selected, you do not need to apply explicit data type conversions for the subset of Unity Pro implicit conversions in this table *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | |
| | Modification: | Online possible |
| | Action required: | **Rebuild All Project** (Offline) |

## FBD

This area is used to define the number of rows and columns for Function Block Diagrams (FBD).

| Element | Description | |
|---|---|---|
| **Number of rows** | From 240 to 1440 (steps of 10). | |
| | Modification: | Offline only |
| | Action required: | No build necessary |
| **Number of columns** | From 60 to 360 (steps of 10).<br>The section area (number of rows x number of columns) cannot exceed the maximum size of 86,400 grids: for example, 240 x 360 or 1,440 x 60.<br>The two input fields are impacting one another. For example, the number of rows is determined by the internal calculation 86,400/number of columns (limited to the lower tenth). | |
| | Modification: | Offline only |
| | Action required: | No build necessary |

## LD

| Element | Description | |
|---|---|---|
| **Right-justify coils** | If this check box is selected, coils *(see page 752)* are automatically placed on the right power rail. | |
| | Modification: | Offline only |
| | Action required: | No build necessary |
| **Number of columns** | Define the number of columns in an `LD` section in this box.<br>From 11 to 63.<br>**Note:** These settings are only valid for new `LD` sections and not for existing sections. | |
| | Modification: | Offline only |
| | Action required: | No build necessary |
| **Cells width** | Using these optional buttons you set the width of the cells from one to three standard cell widths. | |
| | Modification: | Offline only |
| | Action required: | No build necessary |

| Element | Description | |
|---|---|---|
| **Single scan edge detection for EBOOL** | When an EBOOL is written only one time (through OFS, HMI, by program or forced by Unity Pro):<br>● If this check box is selected, writing or forcing an EBOOL activates the edge detection for one cycle.<br>● If this check box is not selected, writing or forcing an EBOOL activates the edge detection until the next writing of this EBOOL. | |
| | Modification: | Offline only |
| | Action required: | No build necessary |

## LD → Mixed display

This area is used to define the available area for variable comments lines height, symbol lines height, and address lines height in mixed display mode *(see page 719)*.

| Element | Description | |
|---|---|---|
| **Comments lines height** | Sets the maximum number of comment lines shown in mixed display mode. From 0 to 20 lines. | |
| | Modification: | Offline only |
| | Action required: | No build necessary |
| **Symbol lines height** | Sets the maximum number of symbol lines shown in mixed display mode. From 0 to 20 lines. | |
| | Modification: | Offline only |
| | Action required: | No build necessary |
| **Address lines height** | From 0 to 5 lines.<br>**NOTE:** These parameters are adjustable only on the **Project Settings**. | |
| | Modification: | Offline only |
| | Action required: | No build necessary |

## SFC

| Element | Description | |
|---|---|---|
| **Allow macro sections** | When this check box is selected, the macro sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* can be used in your project. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow multiple token** | When this check box is selected, the sequential control works in the Multi-token *(see Unity Pro, Program Languages and Structure, Reference Manual )* operating mode. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

## SFC → multi-token

The SFC multi-token option is for PL7 migration and to support the GRAFCET specification that used to be supported by Telemecanique.

| Element | Description | |
|---|---|---|
| **Max. # of active steps/section** | This box is only available if the **Allow multiple token** check box activated.<br>The maximum number of active steps per section can be defined (including all their macro sections) in this box.<br>From 1 to 100. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Max. # of active actions/section** | This box is only available if the **Allow multiple token** check box activated.<br>The maximum number of active actions per section can be defined (including all their macro sections) in this box.<br>From 1 to 100. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **SetSteps: keep previous step(s) in active state** | This box is only available if the **Allow multiple token** check box activated.<br>When this check box is selected, all currently active steps remain active when you set new steps (Animation Panel or function block).<br>When this check box is not selected, all currently active steps are deactivated when you set new steps (Animation Panel or function block). | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

| Element | Description | |
|---|---|---|
| **Parallel branches: allow jump in/out** | This box is only available if the **Allow multiple token** check box activated.<br>When this check box is selected, jumps can be made into or out of a parallel sequence, see Jump into a parallel sequence *(see Unity Pro, Program Languages and Structure, Reference Manual )* and Jump out of a parallel sequence *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Allow multiple evolutions by divergence of sequence selection** | This box is only available if the **Allow multiple token** check box activated.<br>When this check box is checked, all transitions of an alternative branch are processed in parallel (x-out-n-select), see x-out-n-select *(see Unity Pro, Program Languages and Structure, Reference Manual )*.<br>When this check box is not checked, all transitions of an alternative branch are processed in sequence (1-out-n-select), see Alternative sequences *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

**NOTE:** Verify that you optimize the number of active steps and active actions at the right number to get the project to fit into the available memory.

## ST

| Element | Description | |
|---|---|---|
| **Allow jump and label** | When this check box is selected, you can use jumps and labels *(see Unity Pro, Program Languages and Structure, Reference Manual )* in ST. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

## LL984

| Element | Description | |
|---|---|---|
| **SKP enabled** | This LL984 setting is enabled only, if the LL984 library setup was installed.<br>When this check box is selected, the L9_SKP function block can be used in your project.<br>**NOTE:** Modifying this setting leads to a reanalysis of the LL984 networks. This is also possible in online mode. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Online) |

## LL984 → Network Display

| Element | Description | |
|---|---|---|
| **Printing View** | Using this list you select **view1** to **view4** defined under items **View 1** to **View 4** (see below).<br>The default view is **view4**.<br>The view you select for the **Printing View** setting is used for printing the LL984 networks. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **Column Width** | Using this field you can enter the column width as a floating point value (**1.0** to **3.0**. For example, a value of 3.0 means a 3 times larger column width.<br>The default width is **1.0**.<br>**NOTE:** Modifying the **Column Width** setting does not lead to the need of a new **Build**. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |
| **View 1** to **View 4** | The LL984 editor can display various information concerning variables attached to coils, contacts, and function block nodes.<br>According to the settings for **View 1** to **View 4** the LL984 editor shows the following information:<br>● Address (for example, %M100)<br>● 984-address (for example, 400100<br>● Variable name<br>● Variable comment<br>● Descriptors 1 to 9<br>● Topological address<br>For further information, refer to *Items of the View 1 to View 4 Settings (see LL984 Editor, Reference Manual, LL984 Specifics)*. | |
| | Modification: | Online possible |
| | Action required: | **Build Changes** (Offline) |

# Maintain Output Links on Disabled EF (EN=0)

## Introduction

The **Maintain output links on disabled EF (EN=0)** project setting allows you define the behavior of link values, connected to output pins of EFs, in case an EF is called conditionally (EN input connected).

The default value of this project setting for new applications is deactivated (unchecked).

**NOTE:** After changing this setting you have to make a **Rebuild All Project** of the application.

**⚠ WARNING**

**UNEXPECTED EQUIPMENT OPERATION**

Check the behavior of the application when using the Maintain output links on Disable EF function.

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

**NOTE:** The behavior of the application will change after modifying this project setting that has an impact on the code execution.

## Changing this Project Setting

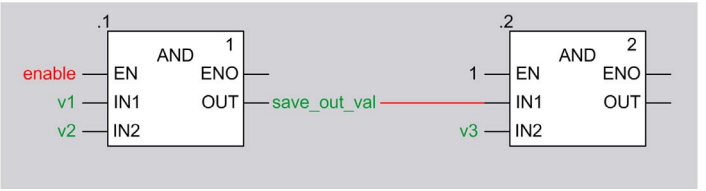
You can change the **Maintain output links on disabled EF (EN=0)** project setting in Unity Pro under **Tools** → **Project Settings** → **Program** → **Languages** → **Common**.

## Behavior in Case of Deactivated Project Setting

In case of a falling edge of EN input, the values of output variables stay unchanged, whereas output links are reset to 0.

For the example shown in the graphic below:

- EN -> 0
- save_out_val = 1
- link = 0 (red)

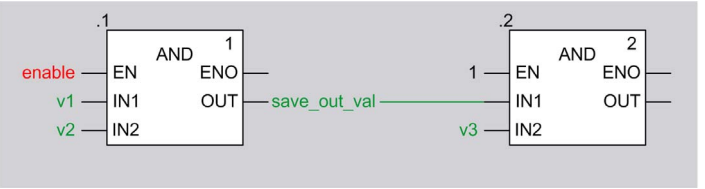


## Behavior in Case of Activated Project Setting

In case of a falling edge of EN input, the values of output variables and output links stay unchanged.

For the example shown in the graphic below:

- EN -> 0
- save_out_val = 1
- link = 1 (green)




## Compatibility Unity Pro 4.0 and 4.1 or later

The following table describes the compatibility of Unity Pro 4.0 (or earlier) and Unity Pro 4.1 or later regarding:

- transferring from PLC to PC (upload)
- opening XEF file on PC (import)
- opening STA file on PC (archived project)

The compatibility depends on whether the project setting **Maintain output links on disabled EF (EN=0)** is activated or deactivated.

| Prerequisite | Unity Pro ≤ 4.0 | Unity Pro 4.1 or later |
|---|---|---|
| Generated with Unity Pro ≤4.0.<br>Project setting not available in Unity Pro 4.0. | possible | possible |
| Generated with Unity Pro 4.1 or later for an application 4.0.<br>Project setting deactivated. | possible (1) | possible |
| Generated with Unity Pro V4.1 or later for an application V4.0<br>Project setting activated. | not possible | possible |
| Generated with Unity Pro V4.1 or later for an application V4.1 or later, using M340 V2.1 PLC, …)<br>Project setting deactivated. | not possible | possible |
| Generated with Unity Pro V4.1 or later for an application V4.1 or later (using M340 V2.1 PLC, …)<br>Project setting activated. | not possible(2) | possible |

**(1)** importing a 4.1 XEF file into Unity 4.0 may require the external tool XEFDowngrader
**(2)** PLC is not recognized

### Project Settings after Upload, Open XEF, Open STA File

The following table describes the status of the **Maintain output links on disabled EF (EN=0)** project setting after

- transferring from PLC to PC (upload)
- opening XEF (or ZEF) file on PC (import)
- opening STA file on PC (archived project)

| Prerequisite | Unity Pro ≤ 4.0 | Unity Pro 4.1 or later |
|---|---|---|
| Generated with Unity Pro ≤4.0.<br>Project setting not available in Unity Pro 4.0. | not available | not activated |
| Generated with Unity Pro 4.1 or later for an application 4.0.<br>Project setting deactivated. | not available | not activated |
| Generated with Unity Pro V4.1 or later for an application V4.0<br>Project setting activated. | not available | activated |
| Generated with Unity Pro V4.1 or later for an application V4.1 or later, using M340 V2.1 PLC, …)<br>Project setting deactivated. | not available | not activated |
| Generated with Unity Pro V4.1 or later for an application V4.1 or later (using M340 V2.1 PLC, …)<br>Project setting activated. | not available | activated |

In case of uploading, importing or opening a Unity Pro application with Unity Pro 4.1 or later, created with Unity Pro 4.0 or earlier, the **Maintain output links on disabled EF (EN=0)** project setting is deactivated.

After a **Rebuild All Project** and download the behavior is the same as the application had in the environment of the earlier version of Unity Pro.

If you want to use the **Maintain output links on disabled EF (EN=0)** project setting, you must:

- open or upload an application,
- activate the project setting,
- **Rebuild All Project**.

## Undoing EF Deletion

If the **Maintain output links on disabled EF (EN=0)** project setting is activated and an EF is called conditionally, static memory will be allocated for each output pin of the EF, to save the output values during runtime.

In case you delete this EF in the editor the static memory will be released.

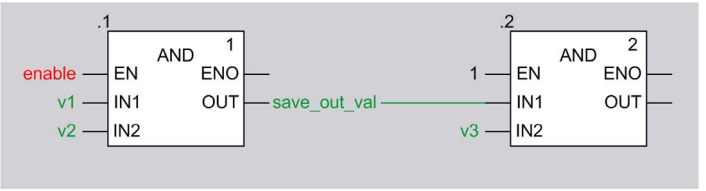
If you perform an UNDO later on, the EF will be re-created and new static memory will be allocated.

After a following **Build Changes**, these mechanisms occur:

- **Maintain output links on disabled EF (EN=0)** project setting deactivated:
  No static memory is assigned for the output pins as maintaining the last link values is not needed.
  - If EN is 0 the link is reset to 0.
  - If EN is 1 the link is written with the value of the OUT.
- **Maintain output links on disabled EF (EN=0)** project setting activated:
  New static memory is allocated for the output pins because the old memory has been released. The link´s new static memory is initialized to 0.
  - If EN is 0 the link's value remains initialized to 0.
  - If EN is 1 the EF is executed and the value of the OUT is written to the link.

Example:

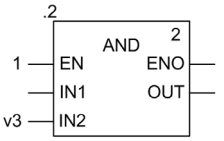
Project setting activated, EN not connected



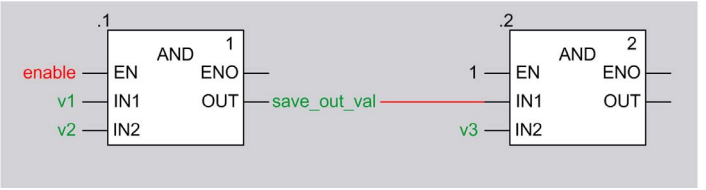

Conditions before **Build Changes**:

- EN = 0
- save_out_val = 1
- link = 1

Delete EF.



Perform UNDO and **Build Changes**.




Result:

- save_out_val = 1
- link = 0

With the next activation of the changed EF the link's new static memory will be written and works as expected on the next deactivation.

This behavior is in principle the same by using other EF types (ADD, XOR…).

## EN Not Connected

If EN is not connected or not displayed (box **Show EN/ENO** not checked), no preparation is done to maintain the last link values.

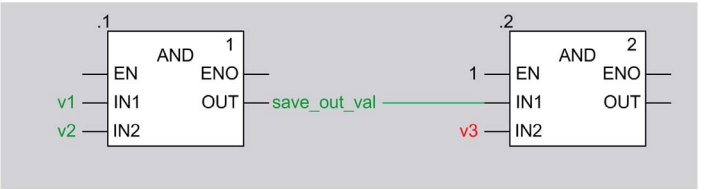
If the EN input is connected later and then **Build Changes** is performed, the following mechanisms occur:

- **Maintain output links on disabled EF (EN=0)** project setting deactivated:
  No static memory is assigned as maintaining the last link values is not needed.
  After falling edge of EN, link is reset to zero, variable save_out_val is unchanged.
- **Maintain output links on disabled EF (EN=0)** project setting activated:
  New static memory is allocated in PLC to maintain the last link values for the output links. This static memory is initialized to 0.
  If EN is 1 after a **Build Changes**, the EF is executed and the output data is assigned to the connected link's static memory.
  The link's static memory will be consistent at the end of the first PLC working cycle after a **Build Changes**.

**NOTE:** If the EN is 0 after the **Build Changes**, the static memory, representing the link, keeps its initialization state (0), which may be different to the value before the download change.

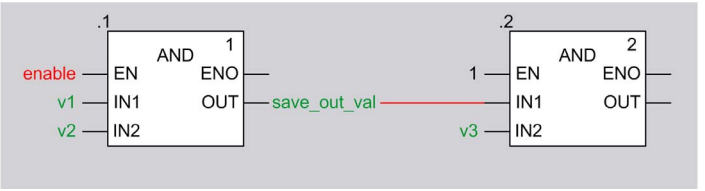
Example:

Option activated, EN not connected



Conditions before **Build Changes**:
- EN of first AND is not connected, i.e. EF AND is invoked
- save_out_val = 1
- link = 1

Connect enable to EN of first AND, set enable to 0, perform **Build Changes**.




Result:
- save_out_val = 1
- link = 0

With the next activation of the changed EF the link's new static memory will be written and works as expected on the next deactivation.

This behavior is in principle the same by using other EF types (ADD, XOR…).

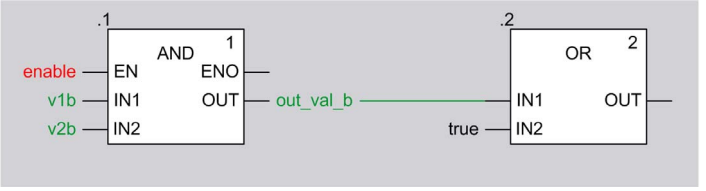
## Use Cases

The following use cases show that
- replacing EFs by other EFs leads to initialization of output link values due to new memory allocation.
- changing output links on not changed EFs leads to maintaining the link values due to using of already allocated memory that belongs to the EFs.

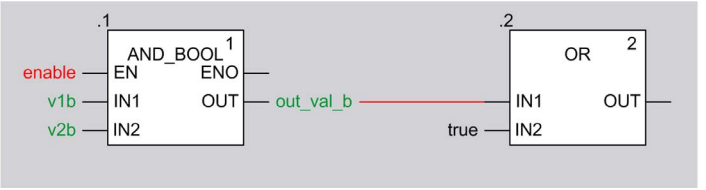
The following use cases apply to all EF types.

## Replacing AND by AND_BOOL

Preconditions
- **Maintain output links on disabled EF (EN=0)** project setting is activated
- variable enable = 0.
- output link = 1 (due to enable was 1 in the history)



Replace AND by AND_BOOL and perform **Build Changes**.




**Result**: output link = 0

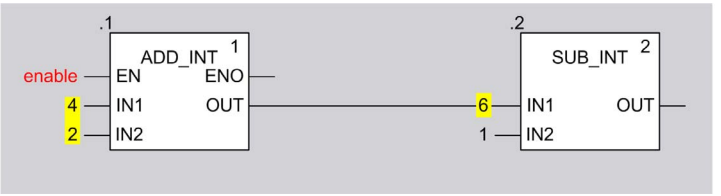
**Reason for output link is 0 after Build Changes**:

- The memory to hold the output values of the deleted EF (AND) has been released.
- Assigning a new EF (AND_BOOL) causes allocation of new memory to hold its output link values.
- The new memory is initialized to 0.

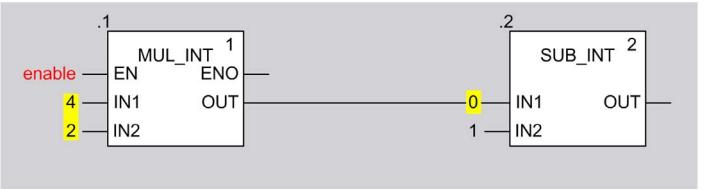
## Replacing ADD by MUL

Preconditions

- **Maintain output links on disabled EF (EN=0)** project setting is activated
- variable enable = 0.
- output link = 6 (due to enable was 1 in the history)




Replace ADD_INT by MUL_INT and perform **Build Changes**.




**Result**: output link = 0

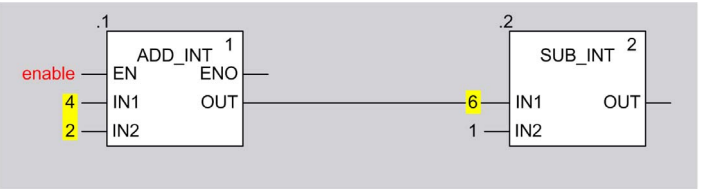
**Reason for output link is 0 after Build Changes**:

- The memory to hold the output values of the deleted EF (ADD_INT) has been released.
- Assigning a new EF (MUL_INT) causes allocation of new memory to hold its output link values.
- The new memory is initialized to 0.

## Deleting Output Link and Performing UNDO

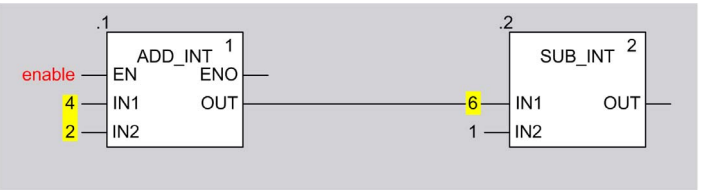
Preconditions

- **Maintain output links on disabled EF (EN=0)** project setting is activated
- variable enable = 0.
- output link = 6 (due to enable was 1 in the history)




Delete output link.

Undo deletion and perform **Build Changes**.




**Result**: output link = 6

**Reason for output link is 6 after Build Changes**:

- Deleting an output link and connecting a new output link to the same EF output pin does not cause the allocating of new memory to hold the values of the output link.
- As this memory belongs to the EF, it will not be released until deletion of the EF.
- In this use case the new link gets the memory of the deleted link (value = 6).

# Configuration Settings for Operator Screens

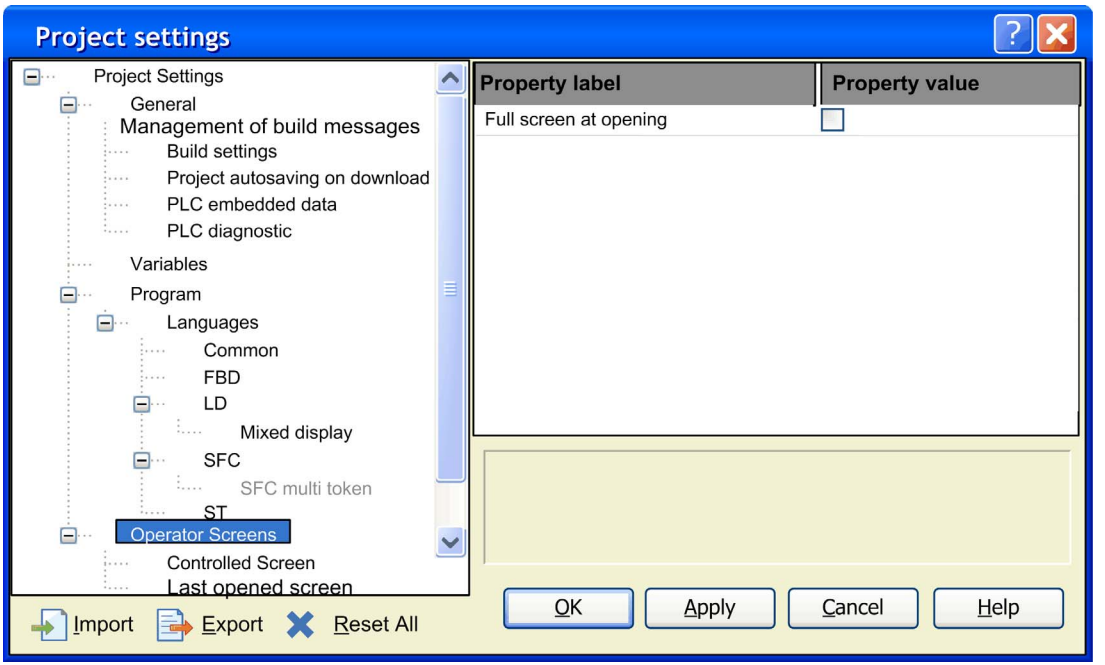
## At a Glance

The configuration section accessible from **Tools** → **Project Settings** → **Operator Screens** is used to:

- Manage the control of screens by the PLC,
- Manage the detection of displayed screens,
- Manage the display or masking of screen numbers in the project browser,
- Manage the display of full screen at opening.

## Representation

Representation of the **Operator Screens** section:



## Description

Elements of the **Operator screens** section:

| Field | Description | |
|---|---|---|
| **Operator Screens** | | |
| **Full screen at opening** | When this box is checked, the operator screens open in **Full screen** mode. The menu bar is replaced by the drawing toolbar. | |
| | Modification: | Online possible |
| | Action required: | No build necessary |
| **Controlled Screen** | | |
| **Variable** | **Variable**: This field is used to enter the screen control variable. In online mode, this variable enables the PLC to select the screen to be displayed. To the right of this field is the  button, which can be used to access the variable instance selection window.<br>**Type**: Type of the selected variable.<br>**Comment**: Comment on the selected variable.<br>**NOTE:** Only variable with type EBOOL, BOOL, byte, INT, or DINT can be selected. | |
| | Modification: | Offline only |
| | Action required: | **Rebuild All Project** (Offline) |
| **Displaying screens controlled via the PLC** | When this box is checked, displaying of screens in online mode is controlled via the PLC. It is the value of the control variable that determines the screen displayed.<br>This setting can also be changed via the shortcuts F2/F3 or their corresponding buttons in the toolbar  when an operator screen is open. | |
| | Modification: | Offline only |
| | Action required: | **Rebuild All Project** (Offline) |
| **Display screen number in the project browser** | When this box is checked, the screen number is displayed to the left of the screen name in the project browser. | |
| | Modification: | Offline only |
| | Action required: | **Rebuild All Project** (Offline) |
| **Last opened screen** | | |

| Field | Description | |
|---|---|---|
| **Variable** | **Variable**: This field is used to enter the DINT type variable containing the number of the last screen opened by the user or the PLC.<br><br>To the right of this field is the ... button, which can be used to access the variable instance selection window.<br>**Type**: Type of the selected variable.<br>**Comment**: Comment on the selected variable.<br><br>**NOTE:** Only variable with type DINT can be selected. | |
| | Modification: | Offline only |
| | Action required: | **Rebuild All Project** (Offline) |

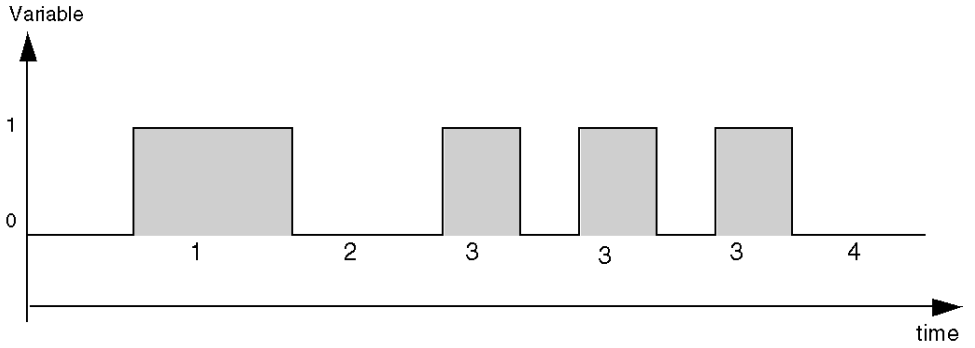
## Control Variable

The control variable enables the PLC to select the screen to be displayed. This variable can be binary or numeric.

## Binary Control Variable

A binary control variable allows controlling the operator screen display with only one button connected to the PLC.

The diagram below gives an example of the trend of the binary control variable:



For each of the steps of the trend diagram, the control variable operates as follows:

| Step | Operation |
|---|---|
| 1 | Its value is equal to 1 for more than 1 second: the screens are highlighted one after another (1 second per screen) according to their position in the project browser. The PLC selects the screens by scrolling through the list from top to bottom. |
| 2 and 4 | Its value is equal to 0 for more than 1 second: the screen currently highlighted in the **Project Browser** (n = current screen) is displayed. |
| 3 | Its value is equal to 1 for less than 1 second: on each pulse, the PLC selects the screen (n+1) in the list of operator screens. The pulses are used to display the screens one by one. |

**NOTE:** As long as this binary control variable has value 1, the operator screens are selected automatically in project browser one after the other. In this case, the context menu cannot be used in the project browser.

### Numeric Control Variable

When the control variable is numeric, its value corresponds to an operator screen number. Therefore, the screen editor displays (if it exists) the screen whose number is equal to the variable value.

# Section 19.2
# Options

## Overview

This section describes the **Tools** → **Options** dialog box tab.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| General Description | 555 |
| General Options | 557 |
| Data and Languages | 559 |
| Connection Options | 561 |
| The Operator Screens Option | 563 |
| Converters | 565 |

# General Description

## Introduction

Functionalities, key shortcuts and specific key-shortcut navigation are used for all Options tabs.

## General Description

The table below describes the general functions used with all options tabs:

| Fields | Description |
|---|---|
| **Import** | This button allows the selection of a Options .XOO file. All values in the file is taken into account immediately. After Import, click on OK or Apply to validate the options.<br>**NOTE:** If a project is opened, the **Custom Library** and **Working Drive** are updated in the next Unity Pro launching. |
| **Export** | This button allows to save the current values of the options in a .XOO file. |
| **Reset All** | This button is used to reset all the options to their default values. |
| **OK** | This button is used to save the modifications and close the current tab. |
| **Apply** | This button is used to save the modifications. |
| **Cancel** | This button is used to cancel without saving the modifications on the current tab. |
| **Help** | This button is used to call the online help. |

## General Navigation

The table below describes the general key shortcut navigation used with all Options tabs:

| Fields | Description |
|---|---|
| **TAB** | This key is used to move from control to another. |
| **ENTER** | This key is used to confirm the current action. |
| **ESC** | This key is used to cancel without saving the modifications in the current tab. |
| **Up, Down, Left and Right Arrows** | These keys are used to navigate within the tree control and the right list box. |

## Specific Navigation

The table below describes the specific key shortcut navigation used with all Options tabs:

| Fields | Description |
|---|---|
| **Space** | This key is used to check / uncheck an element. |
| **F2** | This key is used to activate the current line. |
| **ESC** | This key is used to deactivate the current line. |
| **Up and Down Arrows** | These keys are used to navigate in the list box in the active line. |
| **+ and -** | These keys are used to increase or decrease a value in number field in the active line. |

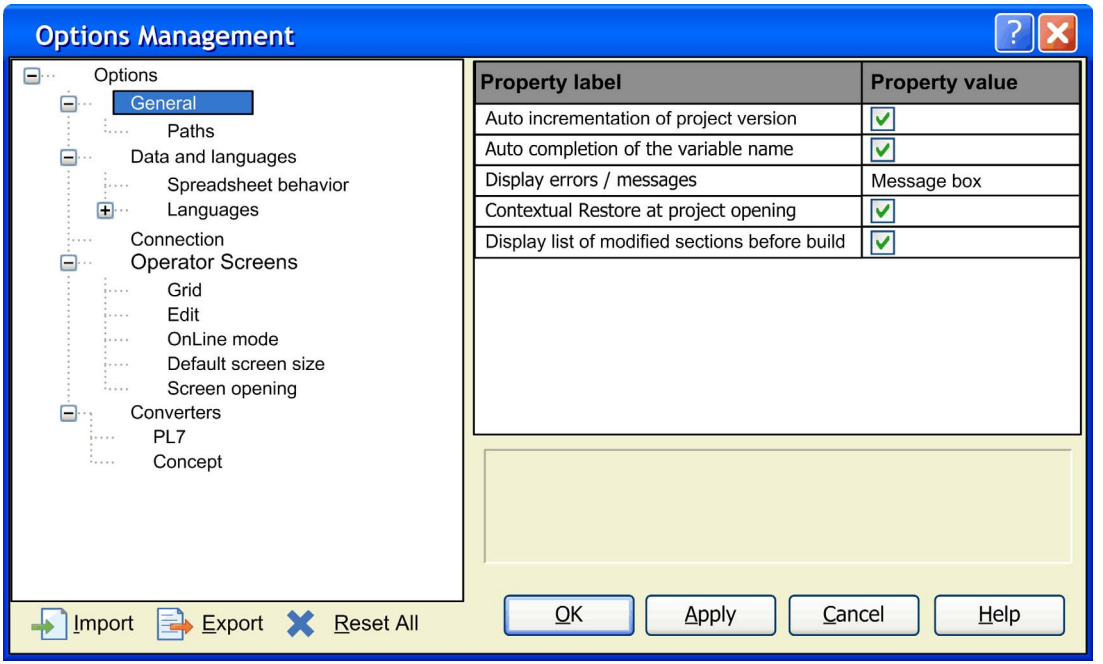
# General Options

## At a Glance

This configuration section, accessible from **Tools** → **Options** → **General**, contains a set of general options.

## General Options Dialog Box

Representation of the **General** section:



## Description

Elements of the **General** section:

| Fields | Description |
|---|---|
| **General** | |
| **Auto incrementation of project version** | When this option is selected, the project version is automatically incremented on each build. |
| **Auto completion of the variable name** | This option is useful when you start the data selection to enter the name of a variable: for example, a pin of a block in FBD.<br>When it is selected, you can enter the first characters of a variable to select one of these that matches with the entered string. If the entered string has been already used before, the auto completion feature completes the string with the last variable that was used. |
| **Display errors / messages** | If **Message box** is selected, all detected error messages are displayed in dialog boxes.<br>If **Output window** is selected, all detected error messages are displayed in the output window. |
| **Contextual Restore at project opening** | If this box is selected, the windows have the same arrangement whereas the project is opened or closed. A .ZTX file is generated, which saves the context. |
| **Display list of modified sections before build** | When this box is checked, the modified sections are listed in the **List of modified sections** dialog box after activating the **Build Changes** command in the **Build** menu. |
| **General → Paths** | |
| **Project path** | Path where the .STU project files are saved or from where they are retrieved when opening. |
| **Import/Export file path** | Path where the import/export source files are kept: for example, .XEF files. |
| **XVM path** | Path where the .XVM file is stored.<br>This file is automatically generated to back up the variables of the project every time this one is transferred to the PLC.<br>The default path is the same as the default **Project path**. |
| **Project settings templates path** | Path where the **Project Settings** templates are located. |
| **Project settings template file** | Project settings template file that is proposed in the **Settings File** during the **New Project** *(see page 83)* creation. |
| **Custom Library path** | Path where the custom libraries are located. Changing the custom library path is taken into account at next open project or new project. |
| **Working drive** | Workspace used by the software. |
| **Log files path** | Path where the workspace is used by the software. |

**NOTE:** The libraries provided by Schneider Electric are located in a Unity Pro system folder. You cannot change the location of these libraries.
To use a previously created custom library after changing the **Custom Library Path**, move the library file to the new location.

# Data and Languages

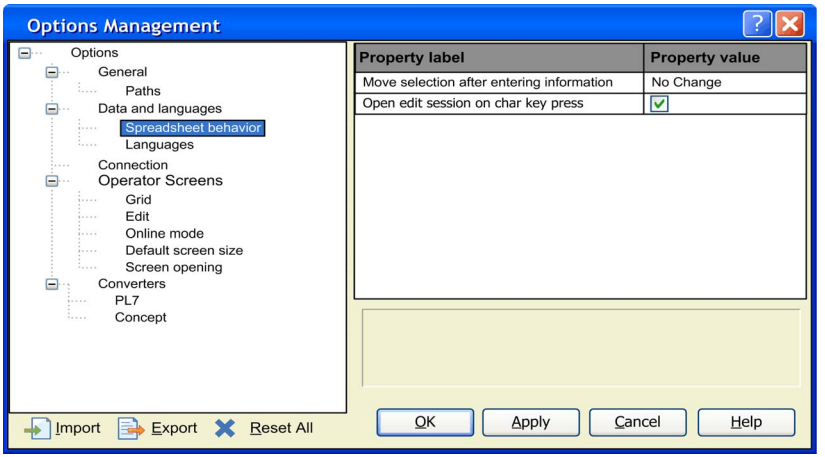
## Introduction

The **Data and Languages** section contains settings for entering data.

**NOTE:** The changes made in this section become active when the respective dialog box (for example, data editor) is closed and then opened again.

## Representation

Representation of the **Data and Languages** section:




Items of the **Data and Languages** section:

| Element | Description |
|---|---|
| **Spreadsheet behavior** | |
| **Move selection after entering information** | This options button allows specifying which neighboring cell becomes active when the **Enter** key is pressed. |
| **Open edit session on char key press** | If this box is checked, the input mode is automatically opened for the currently active cell when you press a character key. The contents of the active cell are overwritten with the new contents. |
| **Languages** | |
| **Automatically assign a variable to a new graphical object** | If this box is checked, the properties dialog box (for example, for contacts, coils, steps, transitions) or the FFB input assistant (for example, for functions or function blocks) is opened. It assigns the formal parameter of the object when you place a graphical object. |

| Element | Description |
|---|---|
| **Reuse windows on refine** | If this box is checked, then:<br>● Only 1 window is used for **Refine** command on DFBs<br>● Only 1 window is used for **Refine** command on actions<br>● Only 1 window is used for **Refine** command on transitions.<br>If this box is not checked, for each **Refine** command on DFBs, actions or transitions, a new window is opened. |
| **LL984** | |
| **Show latched coils as** | Use this list to select letter L (by default) or letter M that is displayed inside a latched coil.<br>**NOTE:** Modifying this setting leads to a refresh of the editor window of all opened LL984 networks. |
| **Reuse 984 editor window when network is opened** | If this box is checked, only 1 window is used for **Open** commands on LL984 networks.<br>If this box is not checked, for each **Open** command on an LL984 network, a new window is opened.<br>**NOTE:** Modifying this setting will only take effect after next time an LL984 editor is opened. |
| **Mnemonics** | The **Property label** column lists all contacts and coils and an assortment of function blocks available for LL984 networks.<br>In the **Property value** column, you can enter a mnemonic for each listed element. A mnemonic can consist of up to 4 alphanumeric signs.<br>Refer to Using Mnemonics *(see page 1156)*. |

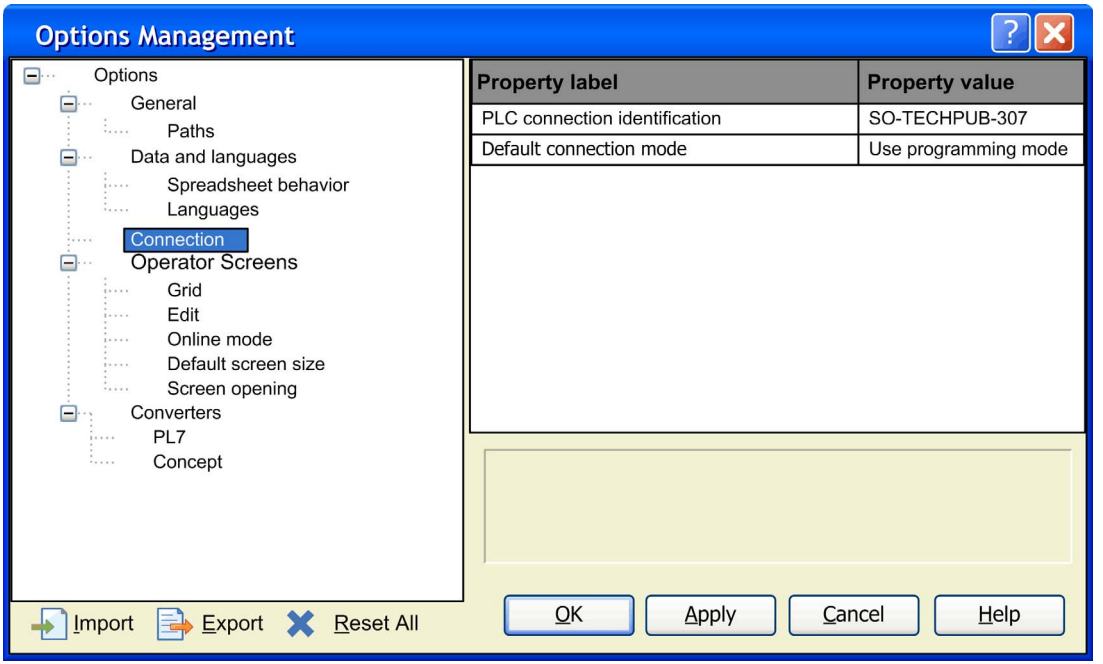
# Connection Options

## At a Glance

This configuration section, accessible from **Tools** → **Options** → **Connection**, contains the options relating to the connection *(see page 92)* of the terminal to the PLC.

## Connection Options Dialog Box

Representation of the **Connection** section:



## Description

Items of the **Connection** section:

| Fields | Description |
|---|---|
| **PLC connection identification** | When several terminals simultaneously access the same PLC, this information can be used to identify which terminal holds the PLC reservation. By default, the network name of the terminal is proposed; this name can be changed. |
| **Default connection mode** | If **Use programming mode** is selected, the terminal connects, by default, in programming mode.<br>If **Ask for connection mode** is selected, the terminal asks, on each connection to a PLC, in which mode the connection is to be made: programming mode or monitoring mode.<br>If **Use monitoring mode** is selected, the terminal connects, by default, in monitoring mode (read-only access).<br>Note: In monitoring mode links between EFs are not animated.<br>If **Mandatory programming mode** is selected, the terminal can only connect in programming mode. If the PLC is reserved, a connection refusal then follows. |

# The Operator Screens Option

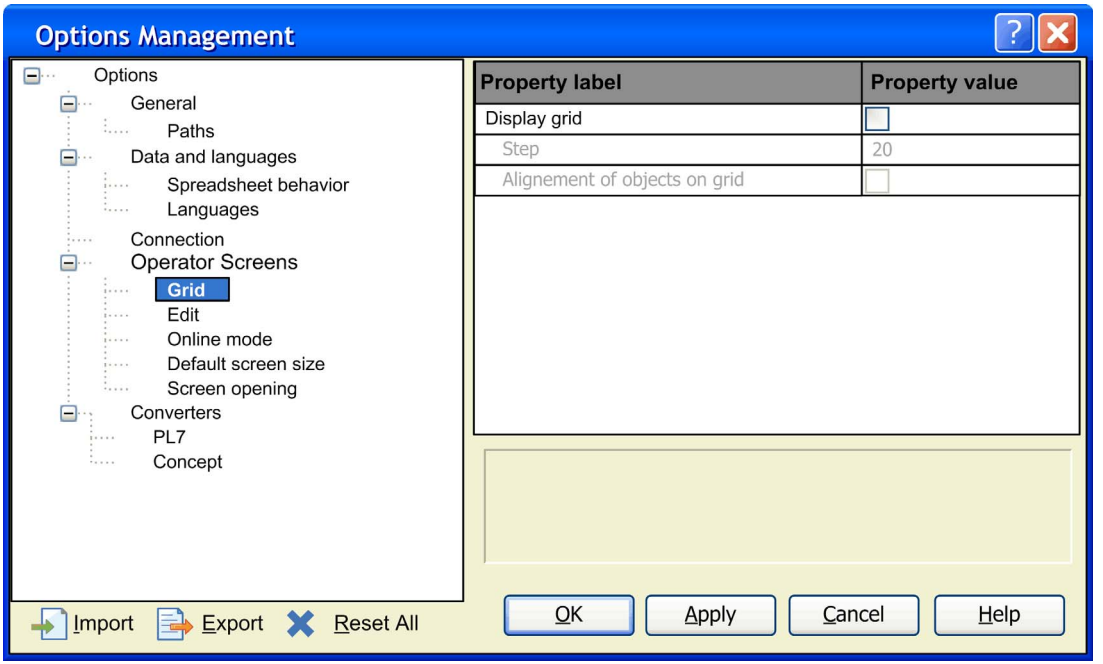
## At a Glance

The configuration section, accessible from **Tools** → **Options** → **Operator Screens**, is used to:

- Enable or prevent use of the magnetic grid,
- Display the mouse position,
- Enable edit in online mode,
- Enable selection of an object in online mode,
- Display or hide the vertical scroll bar,
- Display or hide the horizontal scroll bar,
- Specify the default size of new operator screens.

## Representation

Representation of the register section **Operator Screens**:



Elements of the register section **Operator Screens**

| Field | Description |
|---|---|
| **Grid** | |
| **Display grid** | When this box is checked, the grid is displayed according to the step defined. |
| **Step** | Used to define the step of the grid, i.e. the space between the horizontal and vertical lines of the grid. From 5 to 100 (steps of 5 pixels). |
| **Alignment of objects on grid** | When this box is checked, the objects are automatically positioned on the points of the grid when they are handled (creation, size-modification, movement, etc.).<br>If this box is not checked, the objects are not aligned on the grid. |
| **Edit** | |
| **Display mouse position** | When this box is checked, the position of the mouse is displayed in the status bar. If an object is selected, the display indicates the position of its top left corner, its dimensions and its animation settings (if they exist), see the following example:<br>[x:60,y:180 – cx:100,cy:60][Text][Animation : %MW100] |
| **Vertical scroll bar** | When this box is checked, the vertical scroll bar is displayed when the screen height is larger than the current window. |
| **Horizontal scroll bar** | When this box is checked, the horizontal scroll bar is displayed when the screen width is larger than the current window. |
| **Online mode** | |
| **Enable edit in online mode** | When this box is checked, this authorizes the editing of screens that are not currently controlled by the PLC in online mode (creation, modification and deletion of screens).<br>**NOTE:** If the screens are controlled by the PLC **Description** *(see page 551)*, you cannot edit them in Online Mode even if the **Enable edit in online mode** is checked. |
| **Enable selection in online mode** | When this box is checked, this enables the selection of an object in online mode. |
| **Default screen size** | |
| **Width** | These two fields allow the definition of the default size of new screens. The values entered do not change the size of screens already created. |
| **Height** | |
| **Screen opening** | |
| **Screen opening** | If the Mono-Instance box is checked, when a new screen opens, the latter replaces the previous one in the display window.<br>If the Multi-Instance box is checked, each time a new screen is opened, the screen is displayed in a new window. In online mode, if several screens are opened simultaneously, this can reduce performance. |

# Converters

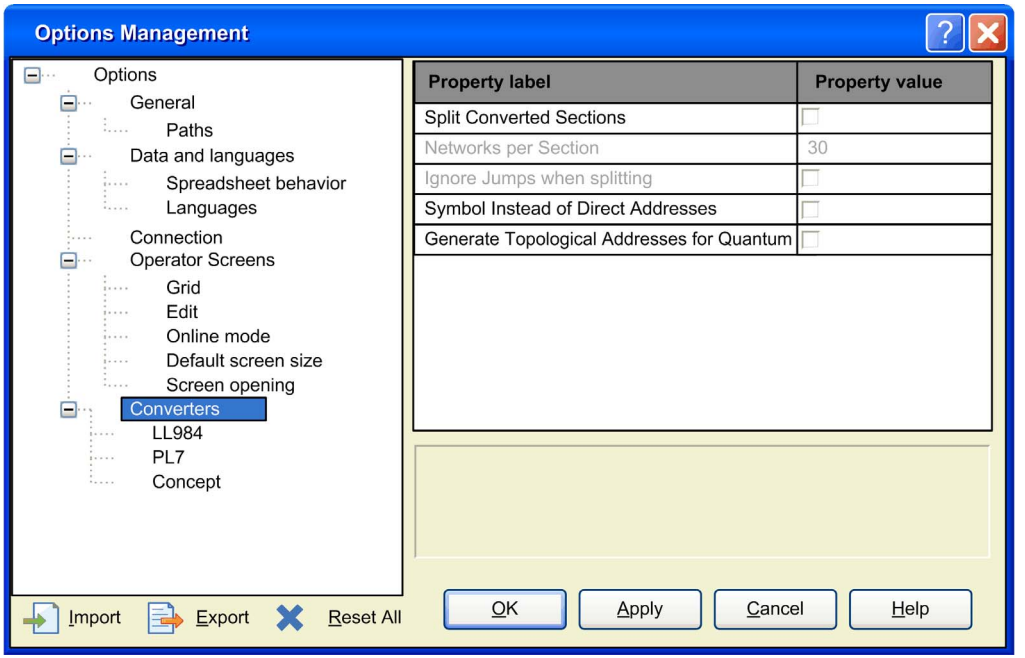
## Introduction

This configuration section, accessed from **Tools** → **Options** → **Converters**, contains options for the conversion of legacy projects to Unity Pro.

The section is only available for Quantum, Premium, and Atrium.

## Representation

Representation of the register section **Converters**:



Elements of the register section **Converters**:

| Element | Description |
|---|---|
| **Converters** | |
| **Split Converted Sections** | This separates ProWORX 32, Modsoft, and Concept LL984 segments into smaller units. Separation is only applied to MAST task sections. Grafcet and Post, FAST task sections and subroutines must be present as a complete unit and cannot be separated into smaller sections:<br>● When this check box is enabled, the options **Networks per Section** and **Ignore Jumps when splitting** are enabled and can be modified.<br>● When this check box is disabled, the options **Networks per Section** and **Ignore Jumps when splitting** are disabled.<br>The separation of segments into smaller units does not take place, if a jump is present in the segment and its target has not yet been reached, and becomes active again after the jump target. |
| **Networks per Section** | Allows the definition of the number of networks in a section.<br>From 1 to 30 networks per section. |
| **Ignore Jumps when splitting** | A section may contain jumps to labels. A jump is located inside the section.<br>The converter takes jumps inside the section into account since it is not allowed to have a jump from one section to another. Depending on how the jump is validated inside the section (active or not), the automatic splitting at network bounds is inhibited (active). The feature of recognizing such jump scopes can be suppressed by this check box. When this check box is enabled, automatic section splitting always takes place regardless of jumps. Therefore many error messages concerning missing jump targets are generated at analyze time. |
| **Symbols Instead of Direct Addresses** | ● When this check box is enabled, discrete variables are converted with their symbols instead of the direct addresses.<br>When there is no symbol in the source application, a symbol is created automatically (for example, %IO.2.3.4 -> IO_2_3_4).<br>● When this check box is disabled, the variables are converted with their direct addresses. |
| **Generate Topological Addresses for Quantum** | ● When this check box is enabled, a topological address is created (for example, %M\2.3\4.5.6).<br>● When this check box is disabled, a **flat address** is generated (for example, %M1234).<br>During the conversion, the Concept addresses are converted into internal flat addresses 1:1. However, if topological addressing is desired, the flat address is searched for in the I/O address tables of the I/O modules. If the associated address is found, the address of the module link point is determined and linked with the address of the I/O module to a topological address.<br>**NOTE:** Flat addressing is only possible for Quantum and for Modicon M340. For Premium, topological addressing is always used regardless of the option. |

| Element | Description |
|---|---|
| **Converters → LL984**<br>These options are present only if LL984 converter is installed. | |
| **Suppress Schedule** | When this check box is enabled, the association of I/O drops (Remote I/O) is suppressed during conversion.<br>This can be done to avoid analyzing errors in Unity Pro. |
| **Suppress Graphs** | When this check box is enabled, networks only containing links are erased during conversion.<br>This can be done to avoid analyzing errors in Unity Pro. |
| **Attach coils to open ends** | When this check box is enabled, coils without assigned variables are positioned at the open ends automatically.<br>This can be done to avoid analyzing errors in Unity Pro. |
| **Generate ConvError Hints** | ● When this check box is enabled, **ConvError objects** are generated in the LD programs during the conversion to draw attention to special issues.<br>● When this check box is disabled, **ConvError objects** are **not** generated in the LD programs. |
| **Include 984 Execution Corrections** | ● When this check box is enabled, the converter modifies the section so that the result in Unity Pro is the same as in ProWORX 32, Modsoft, or Concept.<br>● When this check box is disabled, **no ConvError objects** are generated during the conversion. |
| **Unpowered LD objects to Left Power** | When this check box is enabled, unpowered LD objects are connected to the left power rail automatically with a normally opened contact without assigned variable. |
| **Comment Generation Rule** | When this check box is enabled, is possible to composite a variable comment from different source strings.<br>A blank character is generated between all the different source strings. |
| **Symbol Generation Rule** | When this check box is enabled, is possible to composite a variable comment from different source strings.<br>An underline character is generated between all the different source strings.<br>Possible parameters for the string generation rules are:<br>● **Sx**: short comments while x could be 1…4,<br>● **Dx**: descriptors while x could be 1...9,<br>● **L**: long comment,<br>● **F**: flat address as string without % character.<br>**NOTE:**<br>Variable name source priority:<br>● If Symbol is defined for a State Ram Data, it is used as variable name,<br>● Else, if ISA Symbol is defined for a State Ram Data, it is used as variable name,<br>● Else, a data name is generated with the **Symbol Generation Rule**. |
| **Max. Symbol Length** | Limitation of the generated symbol length. |

| Element | Description |
|---|---|
| **Map Compact to Premium** | ● When this check box is enabled, the Compact modules are converted to Unity Pro Premium modules.<br>● When this check box is disabled, Concept Compact applications are converted to Modicon M340 applications. |
| **Converters → PL7** | |
| **Transition Optimization** | ● When this check box is enabled, in order to improve the performances at execution time, some SFC transitions are converted to variables.<br>● When this check box is **not** enabled, the transitions are converted as they were in PL7. |
| **Step State Explicit Definition** | ● If this check box is enabled, the macro steps that are not used on the PL7 program are declared on the Unity Pro project when the conversion of PL7 application to Unity Pro is done.<br>● If this check box is disabled, the macro steps that are not used on the PL7 program are not converted on the Unity Pro project. |
| **Symbol As Step Name** | ● When this check box is enabled, the PL7 steps %Xi are converted with the PL7 step symbol. For example, on PL7 %X2 with the symbol Motor_1 is converted into Motor_1.<br>● When this check box is **not** enabled, the PL7 steps %Xi are converted on X_i. For example on PL7 %X2 is converted into X_2. |
| **Converters → Concept** | |
| **Variables at empty pins** | ● When this check box is enabled, empty link points are filled with variables created by the converter.<br>● When this check box is disabled, empty link points are **not** filled with variables created by the converter.<br>In Unity Pro, it is necessary to fill provided inputs and outputs for derived data types or I/O parameters (this is not necessary in Concept).<br>If these types are not generic, the converter fills these initially empty inputs and outputs with variables created by the converter. |

| Element | Description |
|---|---|
| **Disable FB Output Multi-Assignment detection** | ● When this check box is enabled, the converter disables the multiple assignment check.<br>● When this check box is disabled, the converter enables the multiple assignment check.<br>Unity Pro and Concept behavior differ on DFB/FB output parameters in FBD or LD sections. This different behavior exists with elementary variables or derived data type components. The converter searches all potential locations where such elementary variables or derived data type components are written and marks these locations with a Converter Error Hint. Then, you can check the affected code concerning different behavior and if necessary modify the application. If the variable is written only once, no Converter Error Hint is generated. |
| **Generate Generic EFs** | ● When this check box is enabled, the converter removes the added data type from the name of the function.<br>● When this check box is disabled, the converter leaves the added data enter the name of the function.<br>There are only a few generic elementary functions in Concept (EFs) for example, MOVE, SEL, MUX. With many other functions, the elementary data type is added to the name of the function.<br>In Unity Pro, many of these functions are used without the elementary data type added to the name (as defined in IEC 61131). The converter removes the added data type from the name of the function.<br>In some cases, the use of generic functions in Unity Pro leads to analytic errors. In these cases, disable the check box. |
| **Separate LD networks** | ● When this check box is enabled, recognized networks are moved vertically, which prevents overlapping.<br>● When this check box is disabled, recognized networks are **not** vertically moved. The original vertical arrangement of the graphics is maintained, but error messages may occur due to overlapping.<br>IEC LD sections contain many independent graphic areas (networks).<br>During the conversion of IEC LD sections, additional columns are added to the networks to avoid undesired automatically generated links in Unity Pro.<br>If the additionally inserted columns were to extend across the entire section, the original graphic would be modified too much. Therefore, the sections are divided in networks during the conversion and additional columns are only inserted for the associated network.<br>Inserting additional columns may cause a network to exceed its maximum section width and it is then wrapped into the next line.<br>If this causes networks to vertically overlap, the overlapping of the logic can lead to undesired automatic links in Unity Pro. |

| Element | Description |
|---|---|
| **Substitute Procedures in ST/IL** | ● When this check box is enabled, the instance name of the Concept call is replaced with the type name.<br>● When this check box is disabled, a **DFB is created**, which accesses the procedure.<br>Some EFBs from Concept are implemented as procedures in Unity Pro without instance names. |
| **Substitute %QD by %MF** | ● When this check box is enabled, **%QD** variables are converted to **%MF** variables.<br>● When this check box is **not** enabled, **%QD** variables are converted to **MW** variables.<br>Variables that are directly addressed in Concept with **%QD** can be initialized floating point constants or dual word constants. When floating point constants appear, this check box should be enabled. |
| **LD Column Break** | The number entered here determines the column after which a network is wrapped to the next column.<br>Inserting additional columns may cause a network to exceed its maximum section width and it is then wrapped into the next line. For more information, see *Wrapping networks that are too wide* in the *Concept Converter* documentation. |

# Section 19.3
## Customize Dialog Box

## Customize Dialog Box

### Call the Dialog Box:

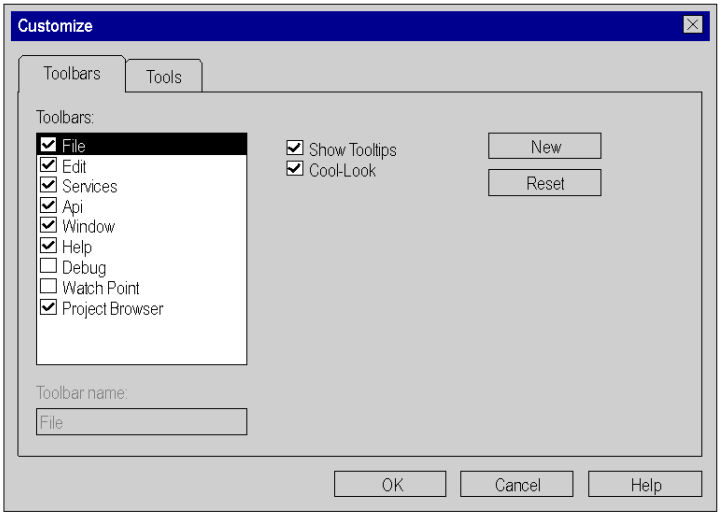
You have the following options to call the dialog box:
- Execute the **Tools** → **Customize** command, or
- Use the **Toolbar** menu command from a toolbar shortcut menu or the toolbar itself.

### Dialog Structure

The dialog box consists of two tabs:
- **Toolbars**
  In this tab, you can show/hide toolbars, create new toolbars, and define the appearance of the toolbars.
- **Tools**
  In this tab, you can define the menu commands for starting external applications.

### Toolbars Tab

Illustration of the **Toolbars** tab:

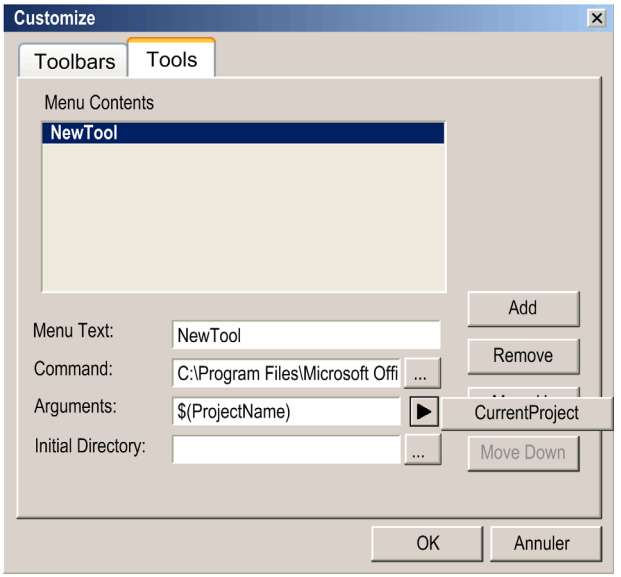


Elements of the **Toolbars** tab:

| Element | Description |
|---|---|
| **Toolbars** | Displays the currently available toolbars.<br>To display a toolbar, activate the respective check box.<br>To hide a toolbar, deactivate the respective check box. |
| **Toolbar name** | Displays the name of the selected toolbar. |
| **Show Tooltips** | When you activate this check box, tooltips are displayed for each toolbar symbol. |
| **Cool Look** | When you activate this check box, the toolbar symbols are displayed in flat form. |
| **New...** | Use this button to open the dialog box for defining new toolbars, see also *Creating your own toolbars, page 1902*. |
| **Reset** | Use this button to restore the original state of the toolbar selected in the **Toolbars** list. |

## Tools Tab

Display of **Tools** tab:



Elements of the **Tools** tab:

| Element | Description |
|---|---|
| **Menu Contents** | Displays the defined menu commands. |
| **Menu Text** | Enter the name of the new command, also see *Starting external applications, page 1896*. |
| **Command** | Enter the path of the file to be executed, also see *Starting external applications, page 1896*. |
| **Arguments** | Enter the arguments to be transferred while launching the application *Starting external applications, page 1896*.<br>By clicking **arrow** → **CurrentProject**, the argument is `$(ProjectName)`.<br>When the application is launched, `$(ProjectName)` is replaced with the complete STU file path of the opened application.<br>**NOTE:** If the application has never been saved, the project file path is undefined. |
| **Initial Directory** | Enter the path to the working directory of the application *Starting external applications, page 1896*. |
| **Add** | Use this button to define a new menu command for starting an external application, see also *Starting external applications, page 1896*. |
| **Remove** | Use this button to delete the selected menu command. |
| **Move Up** | Use this button to move the selected menu command up by 1 position. |
| **Move Down** | Use this button to move the selected menu command down by 1 position. |

# Chapter 20
# Common Function for graphical language

## Overview

This chapter describes the common menus and dialogs of the FBD, LD and SFC editors.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Using Bookmarks in Graphical Languages Sections | 576 |
| Display grid in a FBD/LD/SFC section | 581 |
| Zoom in a FBD/LD/SFC section | 582 |
| Tracking links in a FBD/LD/SFC section | 585 |
| Print Current Section in a FBD/LD/SFC section | 586 |
| Undo and redo changes in a FBD/LD section | 587 |
| Using connectors in a FBD/LD section | 588 |
| Inspect windows in a FBD/LD section | 593 |
| Tooltips for Variables in a FBD/LD/SFC Section | 597 |
| Replace Variables in a FBD/LD Section | 599 |
| Initialize search in a FBD/LD section | 603 |
| Initialize Animation Table in a FBD/LD section | 604 |
| Go to in a FBD/LD section | 605 |
| Go to dialog box in a FBD/LD section | 607 |

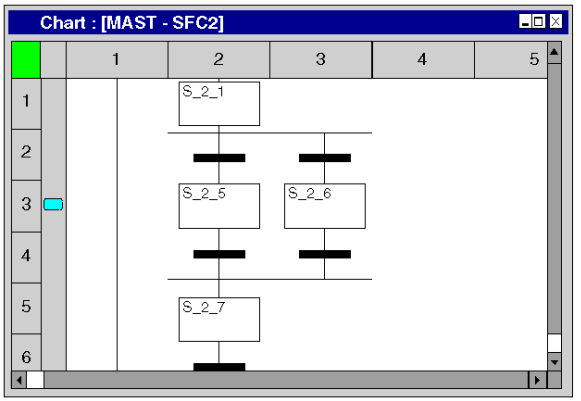
# Using Bookmarks in Graphical Languages Sections

## Introduction

Bookmarks are used to mark positions in the margin (gray column on the left edge of the section/network) and allow the marked positions to be easily found in different language sections/networks.

You can navigate through a list of bookmarks, displayed in the **Tools** → **Bookmarks** dialog.

An example of a bookmark in SFC:




The current bookmark is indicated using a horizontal arrow ( ).

Bookmark are displayed in different colors.

| Created | Enabled/Disabled | Color |
|---|---|---|
| manually | enabled | blue |
| manually | disabled | light gray |
| automatically | enabled | gray |
| automatically | disabled | light gray |

### Setting Bookmarks

You have the following options to set a bookmark:

- Execute the command **Edit** → **Bookmarks >** → **Toggle Bookmark**. (There must not be any current bookmark in the selected area).
  The bookmark is inserted in the margin (gray column on the left edge of the section/network) and is marked as the current bookmark.
- Press the **Ctrl+F2** key combination. (There must not be any current bookmark in the selected area).
  The bookmark is inserted in the margin (gray column on the left edge of the section/network) and is marked as the current bookmark.
  or
- Place the mouse pointer in the margin (gray column on the left edge of the section/network) at the position to be marked and use **Toggle Bookmark** from the context menu for the margin (right mouse click).
  The bookmark is inserted at the mouse pointer and marked as the current bookmark.

### Removing Single Bookmarks

You have the following options to remove a bookmark:

- Select the bookmark to be deleted as the current bookmark (e.g. with **Next Bookmark**) and use the **Edit** → **Bookmark >** → **Toggle Bookmark** command.
- Select the bookmark to be deleted as the current bookmark (e.g. with **Next Bookmark**) and press the **Ctrl+F2** key combination.
  or
- Place the mouse pointer on the bookmark to be removed and use **Toggle Bookmark** from the context menu of the bookmark (right mouse click).
- You can also remove a bookmark using the context menu of the **Bookmarks** dialog.

### Removing All bookmarks

You have the following options to remove all bookmarks in the current section/network:

- Use the **Edit** → **Bookmark>** → **Clear all Bookmarks** menu command.
  or
- Press the **Ctrl+Shift+F2** key combination.
- You can also remove all bookmarks using the context menu of the **Bookmarks** dialog.

### Searching for Bookmarks (Search Down)

You have the following options to search for bookmarks (search down):

- Use the **Edit** → **Bookmark>** → **Next Bookmark** menu command.
- Use the **Edit** → **Go to** → **Bookmark** → **Next** menu command.
  or
- Press the **F2** key.

If a current bookmark is selected, a jump is made to the next one and it is marked as the current bookmark.

The bookmarks are traversed according to their order of appearance in the **Bookmarks** dialog.

### Searching for Bookmarks (Search Up)

You have the following options to search for bookmarks (search up):

- Use the **Edit** → **Bookmark>** → **Previous Bookmark** menu command.
- Use the **Edit** → **Go to** → **Bookmark** → **Previous** menu command.
  or
- Press the **Shift+F2** key combination.

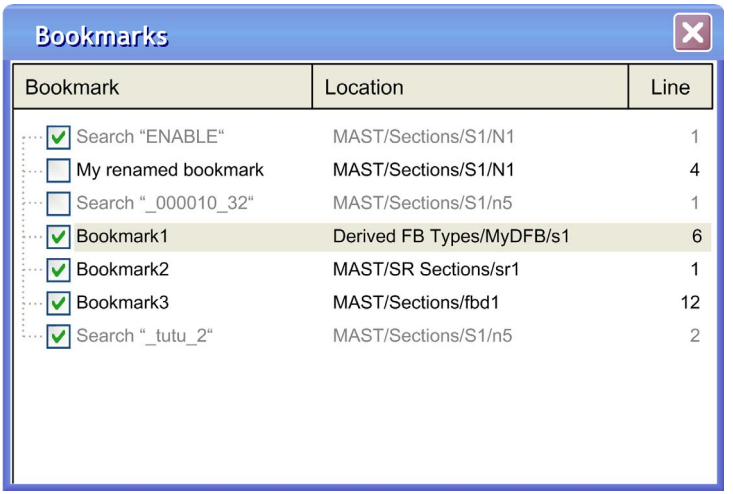
If a current bookmark is selected, a jump is made to the previous one and it is marked as the current bookmark.

The bookmarks are traversed according to their order of appearance in the **Bookmarks** dialog.

### Bookmarks Dialog

To open the bookmarks dialog use **Tools** → **Bookmarks**.

A list of the project´s bookmarks is displayed in the **Bookmarks** dialog.



| Element | Description |
|---|---|
| **Bookmark** column | List of the project´s bookmarks. Automatically created bookmarks (see below) are displayed in gray and can not be renamed. |
| **Location** column | Location of the bookmark (section, network or DFB) referring to the **Project Browser**. |
| **Line** column | Line in the section/network where the bookmark is located. |
| check boxes in the **Bookmark** column | Activate this check box to be able to jump to this bookmark by means of **Next Bookmark**/**Previous Bookmark**.<br>By default for newly created bookmarks the check box is activated. |
| **Delete**/**Delete all** | Right click in the dialog and select from the context menu:<br>● **Delete**: to delete the selected bookmark<br>● **Delete all**: to delete all bookmarks in the project |

Creating bookmarks is done in the language editors.

Deleting and renaming bookmarks can be done in the **Bookmarks** dialog, too.

**NOTE:** Renaming of bookmarks can only be done with created bookmarks. Automatically created bookmarks can not be renamed.

### Bookmark Commands in Language Editors

The bookmark commands in language editors affect the **Bookmarks** dialog as follows:

- **Toggle Bookmark** creates a bookmark, visible also in the **Bookmarks** dialog. It will be appended to the list of bookmarks.
- **Next Bookmark** jumps to the next bookmark in the list of bookmarks
- **Previous Bookmark** jumps to the previous bookmark in the list of bookmarks.
- **Delete Bookmark** deletes the selected bookmark.
- **Delete all Bookmarks** deletes only the bookmarks inside the current section/network.

**NOTE:** The **Bookmark** tab of the **Edit** → **Go To** dialog only displays the bookmarks of the current section.

### Naming Bookmarks

Created bookmarks are named as **Bookmark1**, **Bookmark2** and so on.

In the **Bookmarks** dialog you can rename the bookmarks.

### Automatically Created Bookmarks

Automatically created bookmarks are named as **Search"<Search String>"**.

Automatically created bookmarks are displayed in gray and can not be renamed.

The maximum number of automatically created bookmarks is 30.

Automatic creation of another bookmark will insert the new bookmark as the first one in the bookmark list and will delete the bookmark at the end.

For automatically created bookmarks please refer to Tracing Variables / Addresses *(see page 1249)*.

### Navigating to Bookmarks

In the **Bookmarks** dialog double-click a line in the list of bookmarks to open the respective section/network with the selected bookmark.

If the check box in front of a bookmark line is not activated, you can not jump to this bookmark by means of **Next Bookmark**/**Previous Bookmark**.

### Saving Bookmarks

Bookmarks are saved with your application.

### Deleting Bookmarked Lines

If you delete a section/network or a bookmarked line in a section/network, the corresponding bookmarks are deleted, too.

**Undo** will not restore the bookmarks.

# Display grid in a FBD/LD/SFC section

## Displaying the grid

The editor window has a grid background as default which splits the window into lines and columns.

Use the **View** → **Grid** menu command to show or hide the grid.

The active mode is indicated by way of a check-symbol before the menu command.

# Zoom in a FBD/LD/SFC section

## Introduction

Several Zoom functions are available to provide an optimal display of the elements in the current section (overview of the entire contents or display of details).

## Zoom factor in % (FBD/LD/SFC)

The Zoom factors given in percentages increase or decrease the view of the section in proportion to its normal size. To see the section in its normal size, select **100%**.

The following Zoom factors are available:

- **50%**
  Decreases the section contents to half their normal size.
- **75%**
  Decreases the section contents to three-quarters of their normal size.
- **100%**
  Displays the section contents in normal size.
- **200%**
  Increases the section contents to double their normal size.

The menu command to set the Zoom factor is available:

- via the **View** → **Zoom** menu.
- with right click and menu **Zoom** for the section.

You can see which mode is active by way of a check-symbol before the menu command.

## Custom... (FBD/LD/SFC)

This dialog box has two functions:

- Displaying the current Zoom factor as a percentage
  When the dialog box is opened, the **Zoom %** list displays the current Zoom factor.
- Defining any Zoom factor as a percentage
  You can define the Zoom factor by entering the Zoom factor as a percentage or by selecting a percentage from the list of already used Zoom factors.

The menu commands to set the user-defined Zoom factor is available:

- via the menu **View** → **Zoom** → **Custom...**.
- with right click and menu **Zoom** → **Custom...** for the section.

You can see which mode is active by way of a check-symbol before the menu command.

### Zoom to Fit (FBD/LD/SFC)

For FBD and SFC sections the contents are minimized horizontally and vertically so that the whole section can be seen.

For LD sections the contents are minimized horizontally so that the whole section width can be seen.

If the size of the section is changed, the section contents are automatically adapted to the new size.

The menu command to set the window size is available:
- via the menu **View** → **Zoom** → **Zoom to fit**.
- with right click and menu **Zoom** → **Zoom to fit** for the section.
- using the key combination **Ctrl**+***** (* = multiplication sign of numeric pad)
- using the  symbol.

You can see which mode is active by way of a check-symbol before the menu command.

### Zoom to selection (FBD/LD/SFC)

Decreases or increases the section contents so that the selected element(s) is (are) displayed as large as possible and in focus.

The menu command to zoom to the selection is available:
- via the menu **View** → **Zoom** → **Zoom to selection**.

You can see which mode is active by way of a check-symbol before the menu command.

### Zoom out

To reduce current display by 25 %:
- select the menu **View** → **Zoom** → **Zoom out**.
- right click in the section and select **Zoom** → **Zoom out**.
- press the key combination **Ctrl**+**Minus (-)**.
- using the mouse: press **Ctrl**+**Rotate the mouse wheel**.
- click the  symbol.

### Zoom in

To zoom in on current display by 25 %:
- select the menu **View** → **Zoom** → **Zoom in**.
- right click in the section and select **Zoom** → **Zoom out**.
- press the key combination **Ctrl**+**Plus (+)**.
- using the mouse: press **Ctrl**+**Rotate the mouse wheel**.
- click the  symbol.

## Fullscreen

To display the application window in full-screen mode:

- select the **View** → **Full screen** menu.
- press the key combination **Ctrl**+**F8**.
- click the  symbol.

To end the full screen mode press **Esc** or use the toolbar menu.

# Tracking links in a FBD/LD/SFC section

## Introduction

There are different view functions available to provide optimum display of links (display source, display destination, display entire link).

## Displaying the source of a link

Displaying the source (start point) of a link:

| Step | Action |
|---|---|
| 1 | Select the link whose source you want to display. |
| 2 | Use the **Edit** → **Link** → **Go to Source** menu command.<br>or<br>Use the **Go to Source** menu command from the shortcut menu.<br>**Result:** The focus of the section is changed to display the source (start point) of the link. The current zoom factor remains unchanged. |

## Displaying the destination of a link

Displaying the destination (end point) of a link:

| Step | Action |
|---|---|
| 1 | Select the link whose destination you want to display. |
| 2 | Use the **Edit** → **Link** → **Go to destination** menu command.<br>or<br>Use the **Go to destination** menu command from the shortcut menu.<br>**Result:** The focus of the section is changed to display the destination (end point) of the link. The current zoom factor remains unchanged. |

## Displaying the entire link

Displaying the entire link:

| Step | Action |
|---|---|
| 1 | Select the link that should be displayed in its entirety. |
| 2 | Use the **Edit** → **Link** → **Zoom** menu command.<br>or<br>Use the **Zoom** menu command from the shortcut menu.<br>**Result:** Decreases or increases the section contents so that the selected links are displayed as large as possible and in focus. |

# Print Current Section in a FBD/LD/SFC section

## Introduction

There are the following ways to print the current section:

- Print the section using the menu command
- Print the section using the print documentation

## Print the section using the menu command

Carry out the following steps to print the current section using the menu command:

| Step | Action |
|---|---|
| 1 | Open the print dialog using:<br>● the menu command **File** → **Print**,<br>● or the key combination **Ctrl**+**P**<br>or<br>● the symbol  . |
| 2 | Press **OK** to start printing the current section. |

## Print the section using the print documentation

Carry out the following steps to print the current section using the print documentation:

| Step | Action |
|---|---|
| 1 | Double-click on **Documentation** in the project browser.<br>**Result:** An overview of your system is open, see also *Documentation, page 1695* |
| 2 | Select **Program** → **Tasks** → **MAST/FAST/AUX** → **...** to print the section. |
| 3 | Select the menu command **Print** from the sections context menu. |
| 4 | Press **OK** to start printing the current section. |

# Undo and redo changes in a FBD/LD section

## Undo changes

You have the following options to undo the recent changes:

- Use the **Edit** → **Undo** menu command.
- Press the **Ctrl**+**Z** key combination.
  or
- Select the  symbol.

One change is undone each time the function is executed.

The function can be executed a maximum of 10 times.

## Redo changes

Changes that have been undone using **Undo** can be restored:

- Use the **Edit** → **Redo** menu command.
- Press the **Ctrl**+**Y** key combination.
  or
- Select the  symbol.

One undone change is restored each time the function is executed.

The function can be executed a maximum of 10 times.

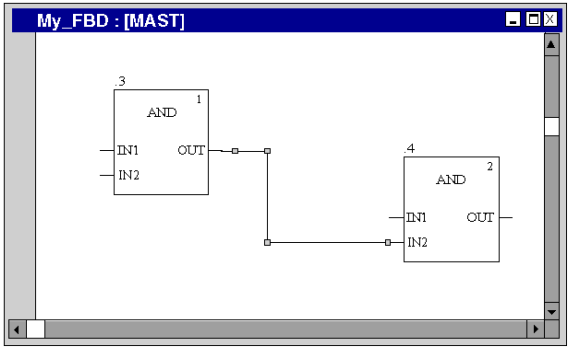
# Using connectors in a FBD/LD section

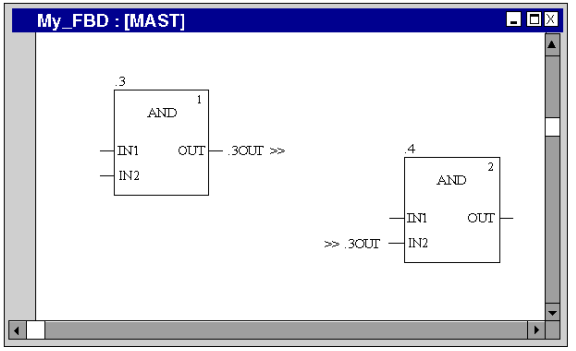
## Introduction

To avoid links overlapping each other, they can also be represented in the form of connectors, see also section Links *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

## Displaying links as connectors

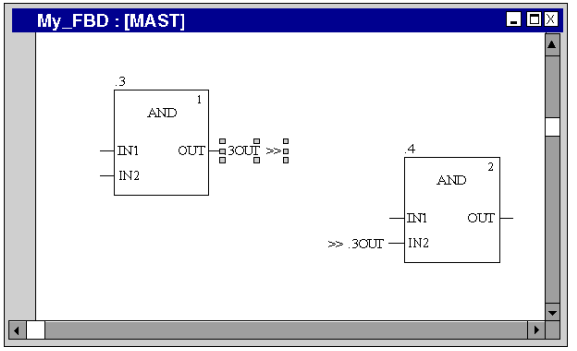
Displaying links as connectors:

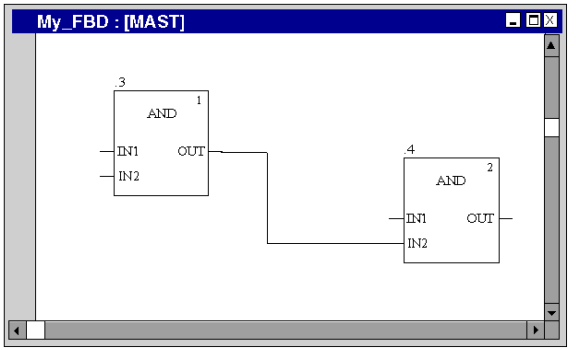
| Step | Action |
|---|---|
| 1 | Select the desired link.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | Use the **Edit → Link → Show as Connector** menu command.<br>or<br>Use the **Show as Connector** menu command from the shortcut menu.<br>**Result:** The link is displayed as a connector.<br>Example:<br>  |

## Displaying connectors as links

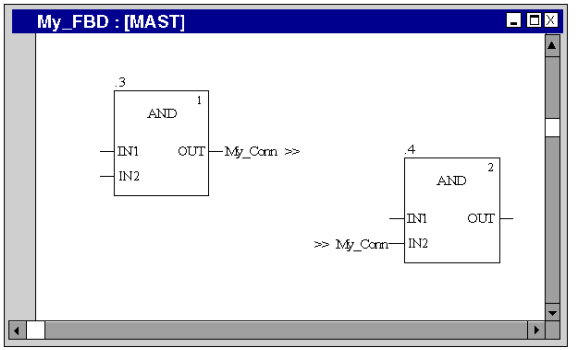
Displaying connectors as links:

| Step | Action |
|---|---|
| 1 | Select the desired connector.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | Use the **Show as Link** menu command from the shortcut menu.<br>**Result:** The connector is displayed as a link.<br>Example:<br>  <br><br>**Note:** Although the connector is now displayed as a link, the name of the connector is still occupied and cannot be used for another link. If the connector name is used for another link, the connector must be explicitly deleted, see *Deleting connectors, page 592*. |

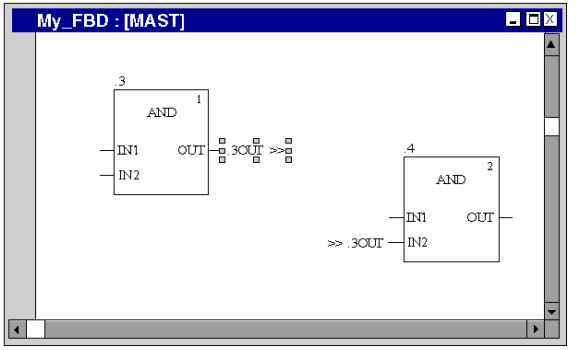
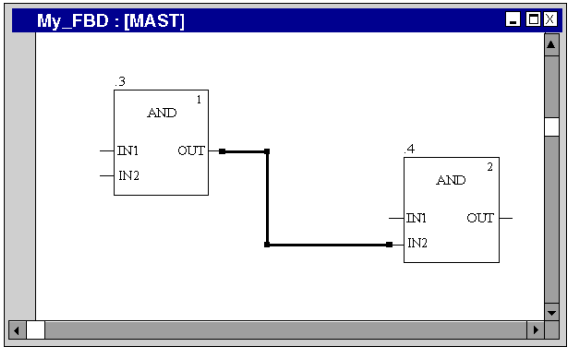
## Renaming connector

Renaming connector:

| Step | Action |
|---|---|
| 1 | Double-click the target connector (source).<br>**Result:** The editor mode is activated. |
| 2 | Enter the new name.<br>**Note:** The name must be unique within the current section and may not already be used by a connector, this also applies to connectors that are to be displayed as links. To enable an occupied connector name, the connector must be explicitly deleted, see *Deleting connectors, page 592*.<br>Confirm the entry using **Enter**.<br>**Result:** The connector (source and target) is renamed.<br>Example:<br>  |

## Deleting connectors

Deleting connectors:

| Step | Action |
|---|---|
| 1 | Select the desired connector.<br>Example:<br>  |
| 2 | Use the **Show as Link** menu command from the shortcut menu.<br>**Result:** The connector is displayed as a link.<br>Example:<br>  |
| 3 | Use the **Delete Connector** menu command from the shortcut menu.<br>**Result:** The connector is deleted. |

# Inspect windows in a FBD/LD section

## Introduction

In offline mode, the name, address (if available) and the comment (if available) of the associated variable are displayed in inspect windows.

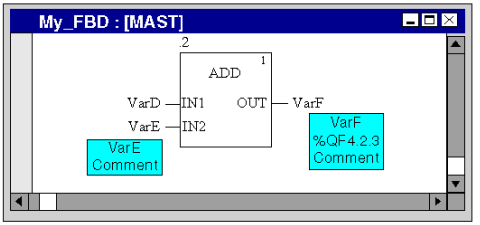
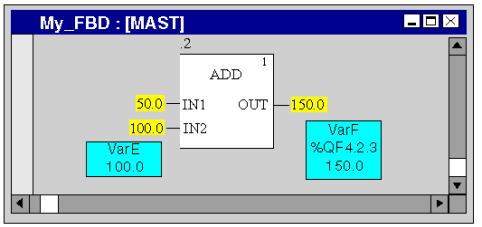
In online mode, the name, address (if available) and the current value of the variable are displayed in inspect windows.

The current values can be displayed:

- Decimal, e.g. `15`
- Binary, e.g. `00001111`
- Hexadecimal, e.g. `F`
- Floating point, e.g. `15.4`
- Exponent, e.g. `1.3e+003`
- Literal, e.g. `'otto'`

## Generating an inspect window

Generating an inspect window:

| Step | Action |
|---|---|
| 1 | **In FBD:** Select the desired FFB pin.<br>**In LD:** Select the desired contact, the desired coil or the desired FFB pin. |
| 2 | You have the following options to generate inspect windows:<br>● Use the **Edit → New → Inspect window** menu command.<br>● Use the **Place Inspect** menu command from the shortcut menu.<br>● Press the **F9** key.<br>or<br>● Select the  symbol.<br>**Result:** The inspect window is generated.<br>Example (offline):<br>  <br><br>Example (online):<br>  <br> |

## Resizing

Position the mouse pointer on the right or left edge of an inspect window (the correct position is indicated by the ↔ or ↔ mouse pointer symbol), click the left mouse button and drag the mouse right or left.

## Changing the position

Position the mouse pointer on the inspect window, click the left mouse button and move the inspect window to the target position.

## Anchoring the position

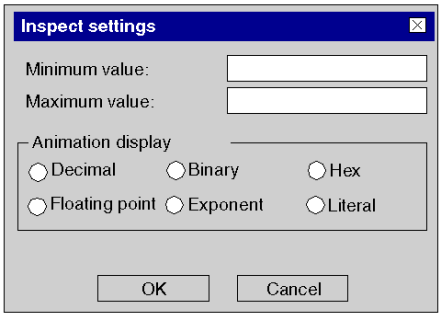
You can anchor all inspect windows of a section to their relative positions within the section (the inspect window stays in its onscreen position even if the page is scrolled), or to their absolute position (the inspect window is scrolled along with the rest of the section contents).

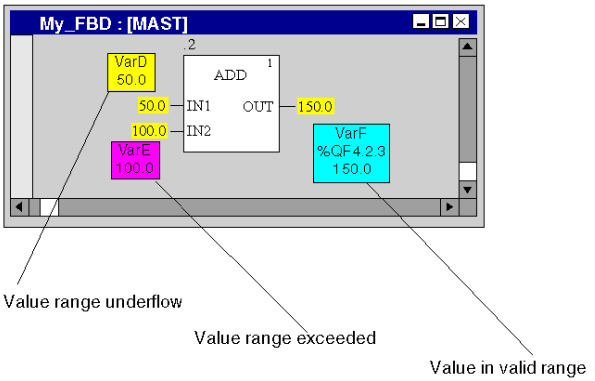
To switch between modes, select the **View** → **Floating Inspect Windows** menu command.

You can see which mode is active by way of a check-symbol in front of the menu command.

## Setting a zone monitor

Setting a zone monitor:

| Step | Action |
|---|---|
| 1 | Position the mouse pointer on the inspect window. |
| 2 | Use the **Settings** menu command from the shortcut menu.<br>**Result:** The properties dialog box for the inspect window is opened.<br>Example:<br>  |
| 3 | Enter a minimum and a maximum value for the variable to be monitored. |
| 4 | Enter the format of the variable displayed in the section. |

| Step | Action |
|---|---|
| 5 | Confirm the entries with **OK**.<br>**Result:**<br>● If the current value is within the defined range, the inspect window is displayed in cyan.<br>● If the current value is outside the defined range, the inspect window is displayed in yellow.<br>● If the current value is above the defined range, the inspect window is displayed in magenta.<br>Example:<br>  <br> |

## Hiding inspect windows

To hide and show the inspect windows select the **View** → **Hide Inspect Windows** menu command.

You can see which mode is active (hide inspect windows) by way of a check-symbol in front of the menu command.

## Deleting inspect windows

Position the cursor on the inspect window to be deleted and select the **Delete Inspect** menu command from the shortcut menu.

# Tooltips for Variables in a FBD/LD/SFC Section

## Introduction

In offline and online mode the following information of a variable can be displayed by tooltip:

- type
- name
- address (if available)
- comment (if available)

Information is displayed for variables and expressions associated to:

- pins of FFBs (FBD/LD)
- contacts, coils, operation and comparison blocks (LD)
- transitions (SFC)

## Concatenated Comments

To show concatenated comments of DDT expressions, set the option **Display complete comments from structure element** via **Tools** → **Project Settings** (for the current project) or via **Tools** → **Options** (for future projects).

Example, when the option is set.

| DDT expression | Tooltip |
|---|---|
| ddt1.b1 | ddt1.b1<br>DDT1 : ddt1 @ %MW100 \| 'Comment for ddt1'<br>BOOL : ddt1.b1 @ %MW100 \| 'Comment for ddt1.Comment for b1' |
| ddt2.ddt1.b1 | ddt2.ddt1.b1<br>DDT2 : ddt2 @ %MW200 \| 'Comment for ddt2'<br>BOOL : ddt2.ddt1.b1 @ %MW200 \| 'Comment for ddt2.Comment for nested ddt1.Comment for nested b1' |

Example, when the option is **not** set.

| DDT expression | Tooltip |
|---|---|
| ddt1.b1 | ddt1.b1<br>DDT1 : ddt1 @ %MW100 \| 'Comment for ddt1'<br>BOOL : ddt1.b1 @ %MW100 \| 'Comment for b1' |
| ddt2.ddt1.b1 | ddt2.ddt1.b1<br>DDT2 : ddt2 @ %MW200 \| 'Comment for ddt2'<br>BOOL : ddt2.ddt1.b1 @ %MW200 \| 'Comment for nested b1' |

### Variable/Pin

Position the mouse pointer over a variable (e.g. `b1`) or expression to display type, name, address and comment of the variable.

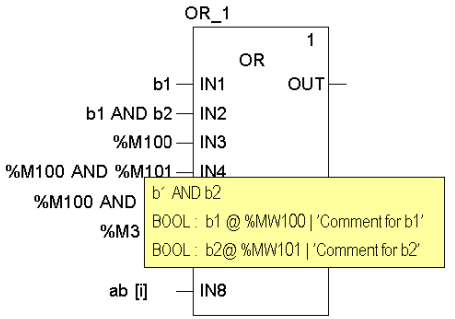
Position the mouse pointer over an FFB pin name (e.g. `IN1`) to display type, name and comment of the pin. In this case the information is taken from the FFB type and not from the FB instance.

### Detected Error Tooltip

There are analyze errors and animation errors.

- Analyze errors are displayed in offline mode in a tooltip, when you position the mouse pointer over a connectable object (e.g. FFB, contact, coil etc.).
- Animation errors are displayed in online mode in a tooltip, when you position the mouse pointer over the `#Err!` value, which is shown in this case as animation value.

### Using the Variable Tooltip

| Step | Action |
|---|---|
| 1 | Position the mouse pointer over the variable or expression. |
| 2 | The tooltip is displayed.<br>Example:<br>  |

# Replace Variables in a FBD/LD Section

## Overview

To replace variables assigned to elementary functions (EF) or to function blocks (EFB, DFB) in FBD/LD sections use the **Replace Variables** dialog box.

To find and replace variables connected to functions and/or function blocks you can use placeholders.

**NOTE:**

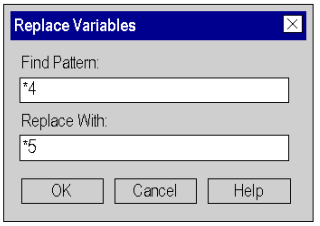
- Structure elements of multi-element variables (DDTs) cannot be replaced.
- If the original variable does not exist, the replacement variable is not possible.

## Dialog Box

You have the following options to display the **Replace Variables** dialog box:

- Select the **Edit** → **Replace Variables** menu command.
- Select the **Replace Variables** command in the context menu (right-click on one or more EF/EFBs in the section).

Dialog box for replacing functions/function blocks using placeholders




## Find and Replace Pattern

The find and replace patterns are applied to all variables connected to the selected functions/function blocks.

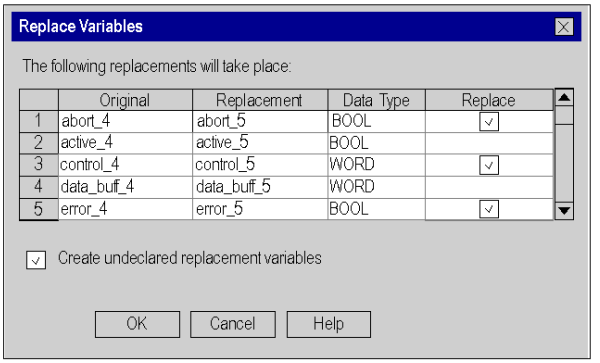
The following placeholders can be used:

- ***** to search/replace any string
- **?** to search/replace any character

Rules for find and replace patterns see below.

## Replace Variables

All variables matching **Find Pattern** will be listed in a dialog box:




Elements of the dialog box

| Element | Description |
|---|---|
| **Original** | displays the names of the found variables<br>The entries in this column are **not** editable. |
| **Replacement** | displays the strings that will replace the names of the found variables<br>The entries in this column are editable. |
| **Data Type** | displays the type of the original variable (not that on of the replacement variable, which could already exist and be of another type)<br>The entries in this column are **not** editable. |
| **Replace** | provides the option to enable/disable the variable replacement via a check box<br>By default **Replace** is enabled for all rows.<br>The entries in this column are editable. |
| **Create undeclared replacement variables** | If this check box is selected, the replacement variables will be created in the Data Editor automatically.<br>If such a replacement variable already exists in the Data Editor, the creation will be skipped.<br>**Note:** If the replacement variable already exists, but has another type than the original one, since creation is skipped the replacement variable may or may not match the type of the connected pin afterwards.<br>**Note:** If the check box is not selected, no variable creation will take place (some of the replacement variables may be undeclared afterwards). |

## Undo/Redo

**Replace Variables** supports Undo/Redo, which means that the original state will be recreated on Undo (of course without deleting the newly created replacement variables from Data Editor) and the state after replacement on Redo.

## Online Mode

**Replace Variables** is supported in online mode.

## Rules for Find and Replace Patterns

Rules for find and replace patterns:

- **?** can be used many times in pattern
- * can be used at maximum 2 times in pattern
- * can be first and last character in pattern only
- no * only and no ** can be used
- no ?* and no *? can be used (always another letter between them)
- numbers of * and ? in both (find and replace) pattern must be equal
- if two * and at least one ? are used the find and replace pattern must be structural identical (same length, same order of normal letters and placeholders)

## Examples

Examples of simple replacements

| Find Pattern: | Found Variable | Replace With: | Variable is Replaced With |
|---|---|---|---|
| abc123jmk | abc123jmk | abc223jmk | abc223jmk |
| abc123jmk | abc123jmk | abc124123jmk | abc124123jmk |

Examples for replacements with constant characters

| Find Pattern: | Found Variable | Replace With: | Variable is Replaced With |
|---|---|---|---|
| ab?1????k | abc123jmk | ab?2????k | abc223jmk |
| abc???12??mk | abc124123jmk | abc???022?4?mk | abc12402234jmk |

Examples for replacements with constant series of characters

| Find Pattern: | Found Variable | Replace With: | Variable is Replaced With |
|---|---|---|---|
| *123jmk | abc123jmk | *223jmk | abc223jmk |
| *123* | abc123jmk | *223* | abc223jmk |
| *123* | abc123jmk | *2234* | abc2234jmk |

Examples of complex replacements

| Find Pattern: | Found Variable | Replace With: | Variable is Replaced With |
|---|---|---|---|
| abc???12??m* | abc124123jmk | abc???022?4?m* | abc12402234jmk |
| *1???mk | abc123jmk | *2???mk | abc223jmk |
| *2??jmk | abc223jmk | *1240?2?4jmk | abc12402234jmk |
| *1?3* | abc123jmk | *2?3* | abc223jmk |

Examples of array replacements

| Find Pattern: | Found Variable | Replace With: | Variable is Replaced With |
|---|---|---|---|
| array[5* | array[5] | array[6* | array[6] |
| array5[5* | array5[5] | array5[6* | array5[6] |
| *5* | array5[5] | *6* | array6[5] |
| array5[i* | array5[i5] | array5[i6* | array5[i65] |
| *5* | array5[i5] | *6* | array6[i6] |

## Not Possible

The following complex replacements are **not** possible

| Entry in text field search pattern | Found variable | Entry in text field Replace With |
|---|---|---|
| *1?3* | abc123jmk | *2?34* |
| *1?3* | abc123jmk | *02?34* |
| *1241?3* | abc124123jmk | *12402?34* |
| *12?1?3* | abc124123jmk | *12?02?34* |

# Initialize search in a FBD/LD section

## Introduction

The **Initialize Search** function transfers the name of the selected element (search text) to the **Element** text box in the **Cross-References** window.

## Selecting the search text

Process the following steps to select the search text:

| Step | Action |
|---|---|
| 1 | Select the element to search for (search text).<br>The following elements can be searched for:<br>● Variable<br>(To search for a variable, select the variable.)<br>● EFB/DFB Instance<br>(To search for an EFB/DFB instance, select the EFB/DFB.) |
| 2 | Use the **Services** → **Initialize Search** menu command.<br>or<br>Select the menu command **Initialize search** from the sections shortcut menu.<br>or<br>Press the **Ctrl**+**U** key combination.<br>**Result:** The search text is transferred to the **Element** text box in the **Cross-References** window. |

# Initialize Animation Table in a FBD/LD section

## Introduction

The **Initialize Animation Table** transfers the selected variables to the Animation table *(see page 1502)*.

## Selecting the variables

Process the following steps to select variables:

| Step | Action |
|---|---|
| 1 | Select the variables to transfer to the Animation table.<br>The following items can be selected:<br>● Variable<br>(To transfer one or more variables, select the variable(s) desired).<br>● FFB<br>(To transfer all variables of an FFB, select the FFB).<br>● Contacts, Coils, Operate and Compare Blocks (LD)<br>(To transfer all variables associated to Contacts, Coils, Operate and Compare Blocks, select these objects).) |
| 2 | Use the **Services** → **Initialize Animation table** menu command.<br>or<br>Select the menu command **Animation Table** from the sections context menu.<br>or<br>Press the **Ctrl**+**T** key combination.<br>**Result:** The variable(s) is/are transferred to the Animation table *(see page 1502)*. |

## Initialize Animation Table

Using **Initialize Animation Table**(**Ctrl**+**T**) you create an animation table with a default name e.g. `Table[FBD Editor - Mixer : [MAST]]`. You can rename the table.

All the variables you selected are listed in the animation table.

Selecting a new set of variables and using **Initialize Animation Table** again, all new variables are added to this animation table (if you did not rename it).

## Initialize New Animation Table

Using **Initialize New Animation Table** (**Ctrl**+**Shift**+**T**) you create a new animation table with an indexed default name e.g. `Table[FBD Editor - Mixer : [MAST]1]`. You can rename the table.

All the variables you selected are listed in the table.

But if you click anywhere in the section and do not select any variable, you create a new empty animation table.

Every time you use **Initialize New Animation Table** you create a new animation table.

# Go to in a FBD/LD section

## Introduction

The **Go to** function is used to jump to:
- a specific location in the current section,
- a specific bookmark in the current section or
- a specific mark in the current section.

## Open the dialog

The following options are available to open the **Go to** dialog box.
- Use the menu command **Edit** → **Go to**,
- select the menu command **Go to** from the sections shortcut menu,
- Press the key combination **Ctrl**+**G**.
  or
- Choose the  symbol.

## Go to a location

Go to a certain position in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 605)* the **Go to** dialog box. |
| 2 | Select the **Position** tab. |
| 3 | Enter the line or column number. |
| 4 | Confirm the entry using the command button **Go to**.<br>**Result:** The desired position is displayed. |

## Go to a bookmark

Go to a certain bookmark in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 605)* the **Go to** dialog box. |
| 2 | Choose the**Actions**tab. |
| 3 | Select a bookmark in the list. |
| 4 | Double click on the selected bookmark or press the **Go to** command button.<br>**Result:** The bookmark is shown and marked as the current bookmark *(see page 576)*. |

## Go to a location

Go to a certain location in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 605)* the **Go to** dialog box. |
| 2 | Select the **Location**tab. |
| 3 | Select a location in the list. |
| 4 | Double click on the selected location or press the **Go to** command button.<br>**Result:** The desired location is displayed. |

# Go to dialog box in a FBD/LD section

## Call the dialog box:

see *Open the dialog, page 605*

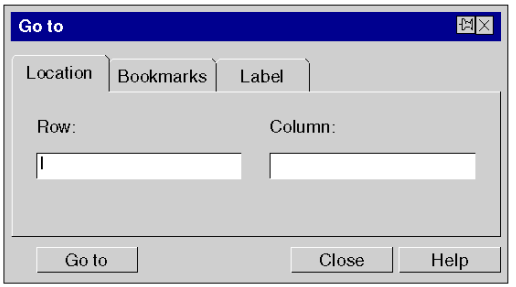
## Call the dialog box

The **Go to**dialog box is made up of three tabs:

- **Location**
  Go to a certain position in the current section using this tab:
- **Bookmarks**
  Go to a certain bookmark in the current section using this tab:
- **Label**
  Go to a certain location in the current section using this tab:

## Position tab

Representation of the tab **Position**:

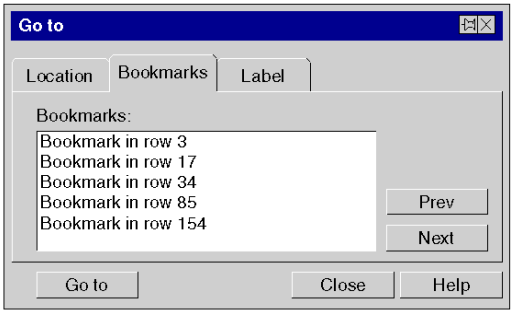



Elements of the **Position** tab

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search. For printed symbols the dialog box remains open until you close it. |
| **Row** | The number of the row to be displayed can be entered in this text box. |
| **Column** | The number of the column to be displayed can be entered in this text box. |
| **Go to** | Display the current position using the button desired position *(see page 605)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

## Register tab Bookmarks

Representation of the**Bookmark** tab:

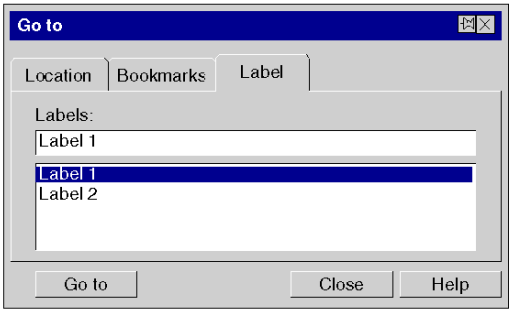



Elements of the tab **Bookmark**:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Bookmark:** | Select the bookmark to be displayed from this list box. |
| **Prev** | If a current bookmark *(see page 576)* is selected, a jump is made to the previous one and the last one is marked as the current bookmark.<br>If no current bookmark is selected, the margin is search from the bottom to the top for the next bookmark, which is marked as the current bookmark.<br>When the start of the section is reached, the search starts again from the bottom. |
| **Next** | If a current bookmark *(see page 576)* is selected, a jump is made to the previous one and the last one is marked as the next bookmark.<br>If no current bookmark is selected, the margin is search from top to bottom for the next bookmark, which is marked as the current bookmark.<br>When the end of the section is reached, the search starts again from the top. |
| **Go to** | Display the current position using the button desired bookmark *(see page 605)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

## Register tab Location

Representation of the tab **Label**:




Elements of the tab **Bookmark**:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Locations:** | Select the location to be displayed from this text box.<br>You can enter the name of the location in the text box with the keyboard or using the drop-down list by clicking with the mouse. |
| **Go to** | Display the current position using the button desired location *(see page 606)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

# Chapter 21
# FBD Editor

## Overview

This chapter describes the menus and dialogs of the FBD editor.

For a description of the syntax for the FBD programming language, please refer to the Function Block Diagram FBD *(see Unity Pro, Program Languages and Structure, Reference Manual )* chapter in the Reference manual.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 21.1 | Creating a program with FBD programming language | 612 |
| 21.2 | Editing FFBs | 630 |
| 21.3 | Returning from a Subroutine or DFB | 670 |
| 21.4 | Calling a Subroutine | 674 |
| 21.5 | Jumps within the Current Section | 678 |
| 21.6 | Definition of the Jump Target (Jump Labels) | 685 |
| 21.7 | Editing Links | 690 |
| 21.8 | Entering Comments | 702 |
| 21.9 | Online functions | 704 |
| 21.10 | FBD Reference Data Type | 705 |
| 21.11 | Export/Import | 706 |
| 21.12 | Customize FBD Editor Colors | 707 |

# Section 21.1
## Creating a program with FBD programming language

### Overview

This section describes the basics for creating a program in the FBD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Structure of an FBD program (function block language) | 613 |
| Creating an FBD Program | 615 |
| Syntax and Semantics Check during programming | 616 |
| Navigating with the keyboard | 617 |
| Selecting objects | 618 |
| Deleting, cutting, copying, pasting and moving objects | 621 |
| Inserting Rows and Columns | 625 |
| Deleting Rows and Columns | 626 |
| Displaying the properties | 627 |
| Refining DFBs and subroutines | 628 |

# Structure of an FBD program (function block language)

## Properties of an FBD program

Properties of an FBD program:

- FBD sections have a grid behind them.
- A grid section consists of ten grid units. A grid unit is the smallest possible space between two objects in an FBD section.
- An FBD section can be configured in number of cells (horizontal grid coordinates and vertical grid coordinates).
- The FBD programming language is not cell oriented but the objects are still aligned with the grid units.
- The execution order is determined by the position of the FFBs in the section (executed from left to right and from top to bottom). If the FFBs are connected to a network using graphical links, the execution order is determined by the signal flow, see also section Execution order of the FFBs *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
  The execution order can be influenced in several ways, see also section Changing the execution order *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
- A syntax and semantics check *(see page 616)* is performed immediately after the statement is entered. The result of this check is displayed in colored text and objects.
- Syntactically or semantically incorrect sections can be saved.

## FBD program objects

The objects of the programming language FBD (Function Block Diagram) help to divide a section into a number of:

- EFs and EFBs *(see page 637)* (Elementary Functions and Elementary Function Blocks),
- DFBs *(see page 637)* (Derived Function Blocks) and
- Procedures *(see page 637)*

These objects, combined under the name FFBs, can be linked with each other by:

- Links *(see page 690)* or
- Actual Parameters *(see page 642)*.

Comments for the logic of the program can be added using text objects *(see page 702)*.

### Edit and view functions

Edit and view functions of the FBD Editor:

- Selecting objects *(see page 618)*
- Deleting objects *(see page 621)*
- Cutting *(see page 621)*, copying *(see page 622)* and pasting *(see page 622)* objects
- Moving *(see page 623)* objects (also between different FBD/LD sections)
- Undo *(see page 587)* and Redo *(see page 587)*
- Using bookmarks *(see page 576)*
- Searching and replacing *(see page 1235)* variables and function blocks and DFB instances.

### Input assistants

Input assistants for the FBD Editor:

- Function input assistants *(see page 652)* for functions, function blocks and procedures

### Online functions

Online functions of the FBD Editor:

- Display the current value *(see page 593)*.
- Setting breakpoints *(see page 1441)*
- Setting watchpoints *(see page 1447)*
- Step by Step *(see page 1444)*

# Creating an FBD Program

## Creating an FBD program

Carry out the following procedures to create an FBD program:

| Step | Action |
|---|---|
| 1 | Creating an FBD section. *(see page 458)* |
| 2 | Place the desired FFB in the section, see also *Calling an FFB via the data selection, page 637*. |
| 3 | Assign the necessary current parameters to the FFB, also see *Assign actual parameters, page 642*.<br>**Note:** As soon as the text is entered several checks are made, such as syntax/semantic error checks, correct spelling of variable names etc. A detailed description of syntax is found in chapter Function Block Diagram FBD *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>The results of the checks are indicated by a color folder, see also *Syntax and Semantics Check during programming, page 616*. |
| 4 | Create a graphical link between FFBs, also see *Placing a Link, page 693*. |

# Syntax and Semantics Check during programming

## Introduction

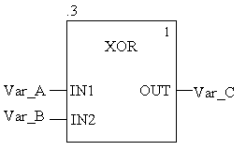
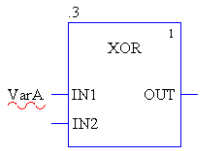
Syntax and semantics check is performed directly when creating the program.

The result of this check is displayed in three formats:

- directly in the program section with colored text
- as tooltip if the cursor is placed on incorrect text.
- In the output window, if **Generate** → **Analyze** is selected.

## Representation

Meaning of colors and labels:

| Color | Description | Example |
|---|---|---|
| Black | Syntax and semantics are correct |   |
| Blue | Possible causes:<br>● Associated variable is not declared<br>● Data type of the variable does not match the data type of the pin |   |
| Red wavy line | Faulty text, e.g. variables which are not declared, variables with an incorrect data type, ... |   |

## Tooltips

If the cursor is placed over an incorrect object or incorrect text, the tooltip is displayed with a brief description of the cause of the error. It will also be displayed in the output window after the analysis.

## Error message in the output window

Error message: `Accessed object is not a function block.`

| Cause of error | Error Correction |
|---|---|
| Name of the function block was already used for a variable. | Rename the already used variable. |

# Navigating with the keyboard

## Using the keyboard

The following keys and key combinations are provided for navigation:

| Key combinations | Motion |
|---|---|
| **Cursor left** | Moves the cursor to the left |
| **Cursor right** | Moves the cursor to the right |
| **Cursor up** | Moves the cursor up |
| **Cursor down** | Moves the cursor down |
| **Ctrl**+**Cursor left** | Moves the cursor one grid point to the left |
| **Ctrl**+**Cursor right** | Moves the cursor one grid point to the right |
| **Ctrl**+**Cursor up** | Moves the cursor one grid point up |
| **Ctrl**+**Cursor down** | Moves the cursor one grid point down |
| **Home** | Shows the beginning of the current line |
| **End** | Shows the end of the current line |
| **Ctrl**+**Home** | Shows the top left cell of the section (the position of the cursor is not affected by this function) |
| **Ctrl**+**End** | Shows the bottom right cell of the section (the position of the cursor is not affected by this function) |
| **Page Up** | Scrolls one page up (the position of the cursor is not affected by this function) |
| **Page Down** | Scrolls one page down (the position of the cursor is not affected by this function) |
| **Ctrl**+**Page Up** | Scrolls one page left (the position of the cursor is not affected by this function) |
| **Ctrl**+**Page Down** | Scrolls one page to the right (the position of the cursor is not affected by this function) |
| **Ctrl**+**Alt**+**Page Up** | Displays the previous section (including via the **View** → **Previous Section**menu). |
| **Ctrl**+**Alt**+**Page Down** | Displays the next section (including via the **View** → **Next Section**menu). |
| **Spacebar** | Selects the object at the cursor position. |
| **Ctrl** + **space bar** | Adds the object at the cursor position to the current selection (multiple selection). |
| **Enter** | Opens the properties dialog box for the selected pin. |
| **Alt**+**Enter** | Opens the properties dialog box for the selected object. |

# Selecting objects

## Select and placement mode

Objects are selected in select mode.

Select mode can be activated using:

- the **Edit** → **Select Mode** menu command
- the  symbol
  or
- the **Esc** key

The  cursor symbol indicates that select mode is active.

## Selecting an object

Selecting an object:

| Using the mouse | Using the keyboard |
|---|---|
| Left-click the object you want to select. | 1. Move the cursor to the object to be selected, see also *Navigating with the keyboard, page 617*.<br>2. Press the **spacebar**. |

## Selecting several objects

Selecting several objects:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Click the left mouse button and keep it pressed.<br>2. Drag the mouse across the objects you want to select.<br>or<br>1. Left-click the first object you want to select.<br>2. Press the **Ctrl** key and keep it pressed.<br>3. Left-click the next object you want to select.<br>4. Repeat these steps until the desired objects are selected. | 1. Move the cursor to the first object to be selected, see also *Navigating with the keyboard, page 617*.<br>2. Press the **spacebar**.<br>3. Move the cursor to the next object to be selected<br>4. Press the **Ctrl**+**spacebar** key combination.<br>5. Repeat these steps until the desired objects are selected. |

## Selecting by rows

Selecting by rows:

| Using the mouse | Using the keyboard |
|---|---|
| Selecting the contents of a row:<br>**1.** Left-click the number of the row you want to select in the vertical ruler.<br>Selecting the contents of several rows:<br>**1.** Left-click the number of the first row you want to select in the vertical ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last row you want to select in the vertical ruler.<br>**Note:** Pressing the **Shift** key makes it possible to combine this procedure with the procedure for selecting columns. | - |

## Selecting by columns

Selecting by columns:

| Using the mouse | Using the keyboard |
|---|---|
| Selecting the contents of a column:<br>**1.** Left-click the number of the column you want to select in the vertical ruler.<br>Selecting the contents of several columns:<br>**1.** Left-click the number of the first column you want to select in the vertical ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last column you want to select in the vertical ruler.<br>**Note:** Pressing the **Shift** key makes it possible to combine this procedure with the procedure for selecting rows. | - |

## Selecting all (entire contents of the section)

Selecting all (entire contents of the section):

| Using the mouse | Using the keyboard |
|---|---|
| Use the **Edit** → **Select All** menu command. | Press the **Ctrl**+**A** key combination. |

## Deselecting objects

Deselecting objects:

| Using the mouse | Using the keyboard |
|---|---|
| Left-click an empty space in the section. | **1.** Move the cursor to an empty space in the section, see also *Navigating with the keyboard, page 869*.<br>**2.** Press the **spacebar**. |

# Deleting, cutting, copying, pasting and moving objects

## Deleting objects

Deleting objects:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 618)* the object to be deleted.<br>**2.** Use the **Edit** → **Delete** menu command. | **1.** Select *(see page 618)* the object to be deleted.<br>**2.** Press the **Delete** key. |
| **Result:** The selected object is deleted. The actual parameters (and the links for FFBs) are deleted together with the object, even if they are not selected explicitly.<br>Please also pay careful attention to the Notes *(see page 661)* on deleting from FFBs **using an execute after** reference. | |

## Cutting objects

Cutting objects:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 618)* the object to be cut.<br>**2.** Use the **Edit** → **Cut** menu command.<br>or<br>Use the menu command **Cut** in the shortcut menu (right mouse button).<br>or<br>Click the  symbol. | **1.** Select *(see page 618)* the object to be cut.<br>**2.** Press the **Ctrl**+**X** key combination. |
| **Result:** The selected object is cut from the section and copied to the clipboard. This is also the case for the actual parameters. Links are also cut. But they are only copied to the clipboard if they (and their partner objects) are selected explicitly.<br>Please also pay careful attention to the Notes *(see page 661)* on deleting from FFBs **using an execute after** reference. | |

The cut object can be inserted *(see page 622)* in any other position (also in another FBD section).

## Copying objects to the clipboard

Copying objects to the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 618)* the object to be copied.<br>2. Use the **Edit** → **Copy** menu command.<br>or<br>Use the **Copy** menu command from the shortcut menu (right-click).<br>or<br>Click the  symbol. | 1. Select *(see page 618)* the object to be copied.<br>2. Press the **Ctrl**+**C** key combination. |
| **Result:** The selected object is copied to the clipboard. This is also the case for the actual parameters. Links are only copied to the clipboard if they (and their partner objects) are selected explicitly.<br>Please also pay careful attention to the Notes *(see page 661)* on copying from FFBs **using an execute after** reference. | |

The cut object can be inserted *(see page 622)* in any other position (also in another FBD section).

## Pasting objects from the clipboard

Pasting objects from the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Use the menu command **Edit** → **Insert**.<br>or<br>Use the menu command **Paste** in the shortcut menu (right mouse button).<br>or<br>Click the  symbol.<br>2. Left-click the target position. | 1. Press the **Ctrl**+**V** key combination.<br>2. Move the cursor to the target position using the arrow keys.<br>3. Press **Enter**. |
| **Note:** This presents the behavior of the **Copy**, **Cut**, **Paste** on a function block instance. It only applies to graphical languages as **FBD** and **LD**.<br>● Using the **Paste** function after a **Copy** of an object:<br>As a result of a **Copy**, a new function block instance (**FBI**) is used. The **Paste** function creates a new FBI whenever it is repeated. Accordingly, the FBI is incremented.<br>● Using the **Paste** function after a **Cut** of an object:<br>As a result of a **Cut**, the same instance of the function is used. The **Paste** function uses the same FBI whenever it is repeated. Accordingly, the FBI is identical.<br>**Note:** Please also pay careful attention to the Notes *(see page 661)* on copying from FFBs **using an execute after** reference. | |

## Moving objects

Moving objects:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 618)* the object to be moved.<br>2. Position the mouse pointer on the selected object. (on one of the selected objects if several are selected).<br>**Result:** The mouse pointer changes its symbol to  .<br>3. Click the left mouse button and keep it pressed.<br>**Result:** The mouse pointer changes its symbol to  .<br>4. Drag the object to the new position and release the mouse button. | 1. Position *(see page 618)* the cursor on the object to be moved.<br>2. Press the **spacebar**.<br>3. Press the **Shift** key and keep it pressed.<br>4. Move the object to the target position using the arrow keys. |
| **Result:** The selected object is moved from its original position to the target position. The object's actual parameters are moved together with the object. FFBs still have their links and they are adjusted to the new position of the FFB.<br>**Note:** The move operation is also possible across sections between different opened FBD sections.<br>**Note:** When objects are moved beyond the visible area of the editor window, automatic scrolling of the window will not begin until you reach the edge of the window with the mouse pointer, not as soon as the objects are moved beyond the edge. | |

## Copying objects via drag & drop

Copying objects via drag & drop:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 618)* the object to be copied.<br>**2.** Position the mouse pointer on the selected object.<br>**Result:** The mouse pointer changes its symbol to  .<br>**3.** Click the left mouse button and keep it pressed.<br>**4.** Press the **Ctrl** key and keep it pressed.<br>**Result:** The mouse pointer changes its symbol to  .<br>**5.** Drag the object to the new position.<br>**Note:** This is also possible across sections between different opened FBD sections.<br>**6.** Release the mouse button.<br>**Result:** A copy of the selected object is pasted at the target position. The object's actual parameters (variable/address) are copied together with the object. A new instance is automatically created for FFBs. Links are only copied if they (and their partner objects) are selected explicitly. Please also pay careful attention to the Notes *(see page 661)* on copying from FFBs **using an execute after** reference. | - |

# Inserting Rows and Columns

## Rules for Inserting Rows

The following rules apply when inserting rows/columns:

- The maximum size of 86,400 grids cannot be exceeded.
- The maximum number of rows is 1440.
- The maximum number of columns is 360.
- The size of one inserted row or column is like the ruler size (that is 10 grids).
- The first row/column selected must not intersect with an object, that has an origin before this row/column.
- The same number of rows/columns are inserted as selected in the vertical/horizontal ruler.
- The new rows/columns are inserted above/left of the first selected row/column (i.e. all objects of the selected rows/columns are moved down/right).
- When rows/columns are inserted within existing links, these links are kept and are adjusted to the new position of the objects.

**NOTE:** The section area (number of rows x number of columns) cannot exceed the maximum size of 86,400 grids, e.g. 240 x 360 or 1,440 x 60. The number of rows and columns will impact one another, e.g. the number of rows is determined by the internal calculation 86,400 / number of columns (limited to the lower tenth).

## Inserting Rows

| Step | Action |
|---|---|
| 1 | Select *(see page 618)* the desired number of rows in the vertical ruler (click the row numbers). |
| 2 | Use the **Insert rows** menu command from the shortcut menu of the vertical ruler or the **Edit** → **Insert rows** command.<br>**Result:** The number of selected rows is inserted above the first selected row. |

## Inserting Columns

| Step | Action |
|---|---|
| 1 | Select *(see page 618)* the desired number of columns in the horizontal ruler (click the column numbers). |
| 2 | Use the **Insert columns** menu command from the shortcut menu of the horizontal ruler or the **Edit** → **Insert columns** command.<br>**Result:** The number of selected columns is inserted in front of the first selected column. |

# Deleting Rows and Columns

## Rules for Deleting Rows

The following rules apply when deleting rows/columns:

- The minimum number of rows (240) respective columns (60) cannot be fallen short.
- The size of one deleted row or column is like the ruler size (that is 10 grids).
- The same number of rows/columns are deleted as selected in the vertical/horizontal ruler.
- When rows/columns are deleted within existing links, these links are kept and are adjusted to the new position of the objects.
- The objects of the selected rows/columns are deleted and the objects of the following rows/columns are moved up/left by the respective number of rows/columns.

## Deleting Rows

Deleting rows:

| Step | Action |
|---|---|
| 1 | Select *(see page 618)* the desired number of rows in the vertical ruler (click the row numbers). |
| 2 | Use the **Delete rows** menu command from the shortcut menu of the vertical ruler or the **Edit** → **Delete rows** command.<br>**Result:** The selected number of rows is deleted. |

## Deleting Columns

Deleting columns:

| Step | Action |
|---|---|
| 1 | Select *(see page 618)* the desired number of columns in the horizontal ruler (click the columns numbers). |
| 2 | Use the **Delete Columns** menu command from the shortcut menu of the horizontal ruler or the **Edit** → **Delete Columns** command.<br>**Result:** The selected number of columns is deleted. |

# Displaying the properties

## Displaying the object properties

You have the following options to display the object properties dialog box:

- Double-click the object.
- Select *(see page 618)* the object and use the **Edit → Properties... dialog box.**.
- Select *(see page 618)* the object and use the **Properties...command.**from the shortcut menu.
- Select *(see page 618)* the object and then press the **Alt**+**Enter** key combination.

When the properties dialog box is called without an object being selected, the section properties dialog box *(see page 461)* is opened.

If several objects are selected, the commands do not work.

## View data properties

You have the following options to display the section properties dialog box *(see page 1197)*:

- using the data editor *(see page 325)*
  **a.** Selecting one or several lines in a list editor
  **b.** Using the shortcut menu select the **Properties**command.
- via the FBD section
  **a.** Selecting one or several element(s) in the FBD section.
  **b.** From the context menu select the command **Data properties** or press **Ctrl** + **Enter**.

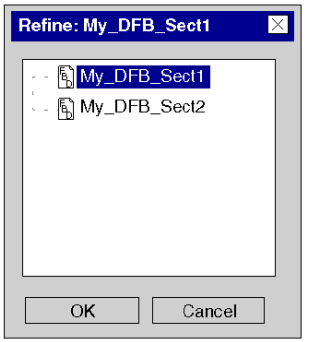
# Refining DFBs and subroutines

## Introduction

This function makes it possible to "look into" a DFB or subroutine. This makes it possible to see the internal state of the DFB/subroutine when animation is used, or to make changes to the logic very quickly.

## Refining DFBs

Process the following steps to refine DFBs:

| Step | Action |
|---|---|
| 1 | Select the DFB. |
| 2 | You have the following options to refine DFBs:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** If the DFB contains only one section, this will be opened automatically. If the DFB contains several sections, a selection dialog is opened.<br>Example:<br>  |
| 3 | Select the desired DFB section.<br>**Result:** The DFB section is opened.<br>**Note:** With nested DFBs, several **refine** processes are possible. |

## Refining subroutines

Process the following steps to refine subroutines:

| Step | Action |
|---|---|
| 1 | Select the subroutine block *(see page 674)*. |
| 2 | You have the following options to refine subroutines:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** The subroutine is opened.<br>**Note:** With nested subroutines several **refine** processes are possible. |

# Section 21.2
# Editing FFBs

## Overview

This section describes the various possibilities to edit elementary functions, elementary function blocks, derived function blocks and procedures in the FBD programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| General information about calling an FFB | 631 |
| Calling an FFB via the FFB Input Assistant. | 633 |
| Calling an FFB via the data selection | 637 |
| Assign actual parameters | 642 |
| Using Public Variables | 654 |
| Expanding Functions | 656 |
| Negating FFB Pins | 657 |
| EN and ENO Show and Hide | 658 |
| Modifying the execution order | 659 |
| Replacing an FFB | 663 |
| Properties dialog box for elementary functions, procedures and sub-routine blocks. | 665 |
| Properties dialog box for elementary and derived function blocks (FBs) | 667 |

# General information about calling an FFB

## Introduction

FFB is the generic term for:

- Elementary Function (EF) *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- Elementary Function Block (EFB) *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- Derived Function Block (DFB) *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- Procedure *(see Unity Pro, Program Languages and Structure, Reference Manual )* (Procedures are an expansion to IEC 61131-3 and must be enabled explicitly in the **Tools** → **Project Settingsdialog box, on the** Language Extensions **tab by selecting the** Allow procedures **checkbox)**.

FFBs can be placed in any free space.

If an FFB is placed in a space that is already occupied by an object, an error message is returned.

## Declaring block instances

Function blocks (elementary and derived) have internal states. If the inputs have the same values, the value on the output can have another value during the individual executions. For example, with a counter the value on the output is incremented.

The instance name serves as a unique identification for the function block in a project.

Instances can be declared:

- By selecting a function block type and its placement in a section.
- By the declaration of the block instance in the FFB input assistant *(see page 1194)* its placement in the section.
- By the declaration of the block instance in the Data editor *(see page 359)*.

**an FFB call.**

FFBs can be called in the following ways:

- Via the FFB Input Assistant... *(see page 633)* menu command
  - Use the **Edit** → **FFB Input Assistant...** menu command.
  - Use the menu command **FFB Input Assistant...** from the shortcut menu.
    or
  - Press the **Ctrl**+**I** key combination.
- Via the data selection *(see page 637)*
  - Use the **Edit** → **Data Selection...**.
  - Use the menu command **Data Selection...** from the shortcut menu.
  - Press the **Ctrl**+**D** key combination.
    or
  - Select the  symbol.
- Via the **Types library browser** per drag & drop
  - Use the **Tools** → **Types Library Manager** menu command.
    or
  - Press the **Alt**+**3** key combination.

# Calling an FFB via the FFB Input Assistant.

## Activating the FFB input assistant

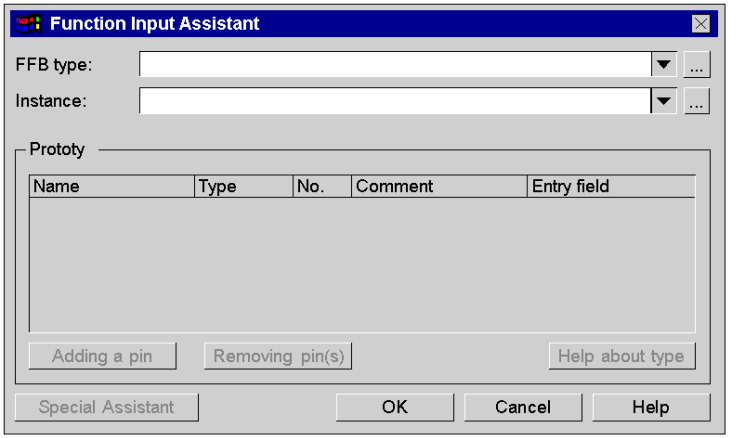
You have the following options to activate the FFB input assistant:

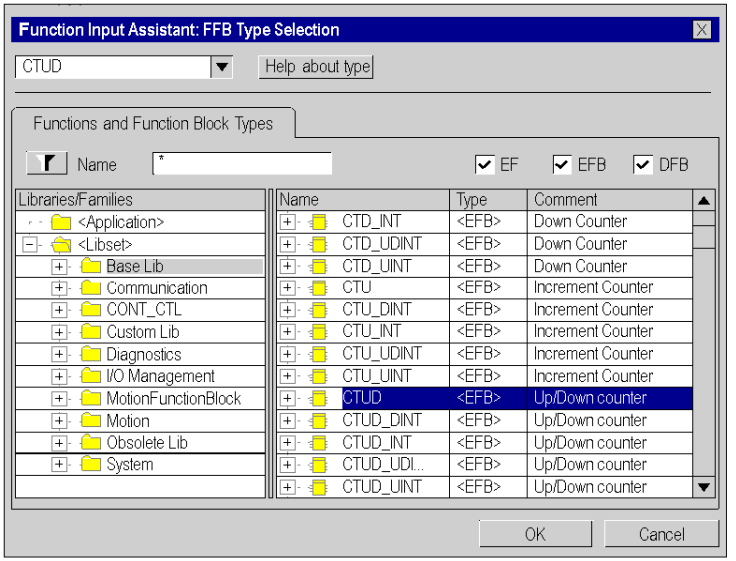
- Use the **Edit → FFB Input Assistant... menu command.** (no object may be selected).
- Use the **menu command** from the shortcut menu (no object may be selected).
  or
- Press the key combination **Ctrl+I** (no object may be selected).

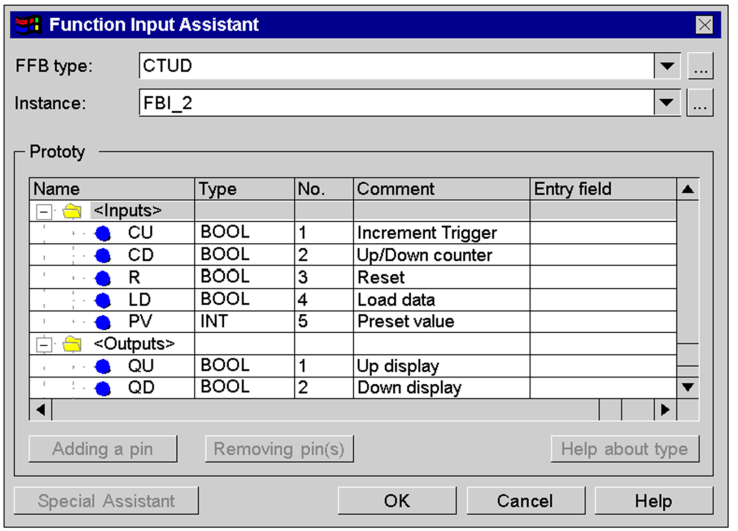
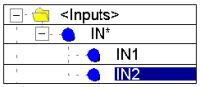
The active FFB placement mode is indicated by the cursor symbol.

## Selecting an FFB type via the FFB input assistant

Process the following steps to select an FFB type via the FFB input assistant:

| Step | Action |
|---|---|
| 1 | From the **FFB input assistant....**from the shortcut menu (right-click) for the FFB input assistant *(see page 633)*section.<br>The FFB input assistant *(see page 1194)* is opened.<br>  |
| 2 | You have the following options to enter the FFB type with the:**FFB Type** text box:<br>● You can enter the FFB type in the textbox:<br>In this case, continue with step 6 of the procedure.<br>● You can select the FFB type from the list of recently used names using the ▼ symbol.<br>In this case, continue with step 6 of the procedure.<br>● Using the button **...** open the FFB selection dialog *(see page 1187)*. |

| Step | Action |
|---|---|
| 3 | Select the library from the **Libraries/Families** column which contains the desired function block.<br>Tip: If you do not know where to find the FFB you are looking for select the **Libset**, entry to see a list of all the FFBs available independently of their libraries. |
| 4 | Select the desired function block from the **Name** column.<br>Example:<br>  |

| Step | Action |
|---|---|
| 5 | Confirm the selection with **OK**.<br>**Result:** The dialog box is closed and FFB input assistant is activated.<br>Example:<br>  |
| 6 | You have the following options to enter the block instance for function blocks in the **instance** text box:<br>● You can accept the name which is suggested.<br>● You can change the name which is suggested. |
| 7 | The number of inputs *(see page 656)*can be increased with some elementary functions.<br>Expandable pins can be recognized from the following display:<br>  <br>To add further pins select the last pin in the structure and press **Add pin**. |
| 8 | To assign the current parameter double-click the **Entry field** cell of the first formal parameter and enter the parameter to be used.<br>You have the following options to enter the variables/addresses:<br>● You can enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the button **...**to open a variable selection dialog box *(see page 1182)*.<br>Assign an actual parameter to all formal parameters of the function block in this way. |

| Step | Action |
|---|---|
| 9 | Confirm the entries using the **OK** button.<br>**Result:** The FFB input assistant is closed and the FFB input mode is activated. The active FFB placement mode is indicated by the  cursor symbol. |
| 10 | To place the FFB , and click the desired location in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The FFB is inserted and a syntax and semantics check *(see page 717)* is performed. |

# Calling an FFB via the data selection

## Set data selection to active

You have the following options to activate the data exchange:

- Use the **Edit → Data Selection... menu command**.
- Use the menu command**Data selection...**from the shortcut menu (right-click) for the section.
- Press the **Ctrl**+**D** key combination.
  or
- Select the  symbol.

**NOTE:** When executing the commands no object can be selected in the section.

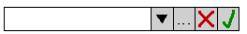
The active FFB placement mode is indicated by the  cursor symbol.

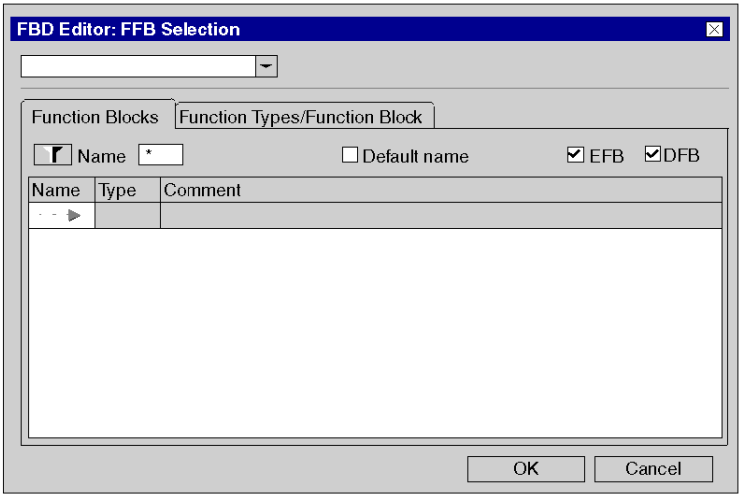
## Note

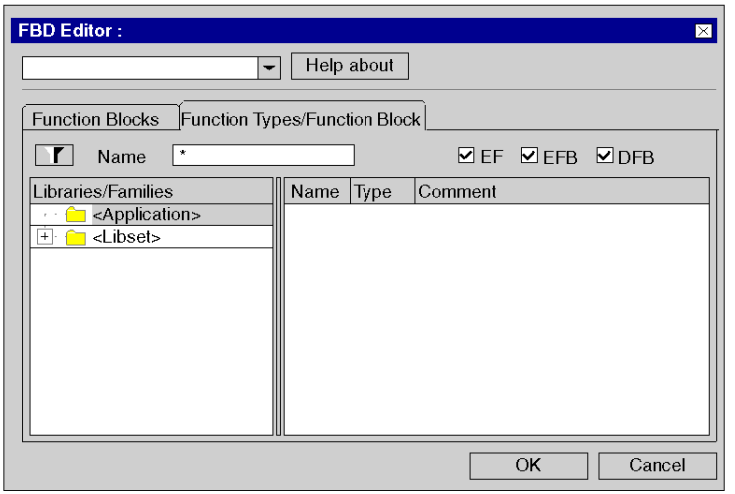
In the **Tools → Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the corresponding input assistant is automatically opened when an FFB is placed. The procedures explained here are used when the checkbox is **cleared**.

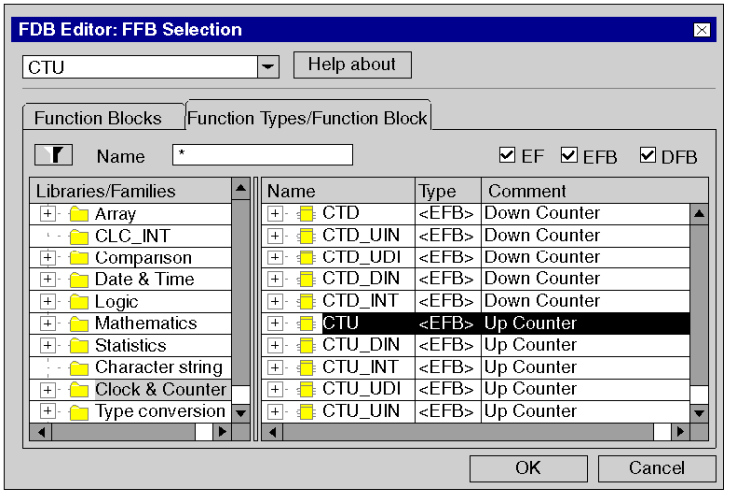
## Selecting an FFB type via the Data Selection

Process the following steps to select an FFB type via the Data selection:

| Step | Action |
|---|---|
| 1 | From the **Data selection....**from the shortcut menu (right-click) for the Data selection *(see page 637)*section.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |

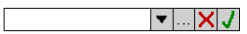
| Step | Action |
|---|---|
| 2 | You have the following options to enter the FFB type:<br>● You can now enter the FFB type and confirm using **Enter**or the ✓button. In this case, continue with step 7 of the procedure.<br>● You can select the name from the list of recently used names using the FFB type▼ symbol and confirm using **Enter**or the button ✓. In this case, continue with step 7 of the procedure.<br>● You can delete your entry by pressing **Esc**or clicking the ✕ button.<br>● Using the button **...you can open** the dialog box for FFB selection *(see page 1184)*.<br>Example of a FFB type selection *(see page 1185)* dialog box:<br>  |

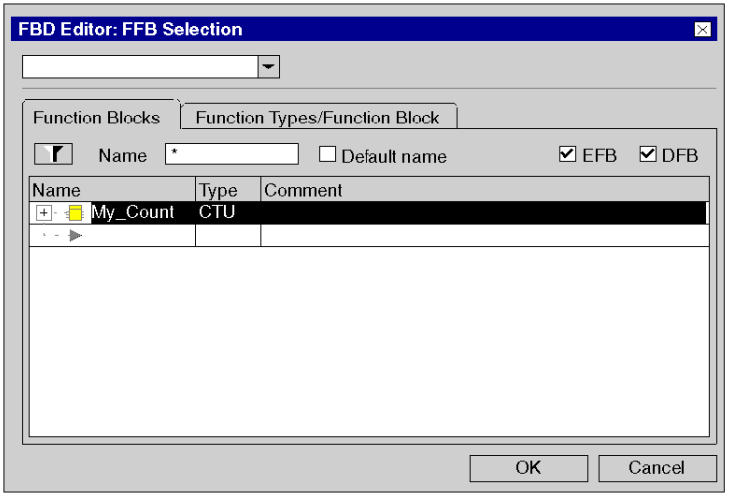
| Step | Action |
|---|---|
| 3 | Select the **Function Types, Function Block Types** tab.<br>**Result:**<br>The tab used for selecting an FFB type is opened.<br>  |
| 4 | Select the library and family from the **Libraries/Families** column which contains the desired function block.<br>**Tip:** If you do not know where to find the FFB you are looking for select the **Libset**, entry to see a list of all the FFBs available independently of their libraries. |

| Step | Action |
|---|---|
| 5 | Select the desired function block from the **Name** column.<br>Example:<br>  |
| 6 | Confirm the selection with **OK**.<br>**Result:** The dialog box is closed and FFB placement mode is activated. |
| 7 | To place the FFB , and click the desired location in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** For the selected FFB type, an instance name is automatically generated (only valid for function blocks), the FFB is inserted and placement mode remains active so that further FFBs can be inserted. To end insert mode press **Esc**.<br>**Note:** The automatically generated instance names can be changed to improve clarity, see also *Managing of instances of data belonging to the family of function blocks (EF), page 359*. |

## Selecting a block instance

Process the following steps to select a block instance that has already been declared:

| Step | Action |
|---|---|
| 1 | Activate FFB placement mode.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |
| 2 | Press the button **...**, to open the FFB selection dialog *(see page 1184)*. |

| Step | Action |
|---|---|
| 3 | Select the **Function Blocks** tab (if it is not already active).<br>**Result:**<br>The tab used for selecting a block instance is opened and the desired selection of block instances *(see page 1184)* is displayed.<br>  |
| 4 | Select the desired block instance from the **Name** column. |
| 5 | Confirm the selection with **OK**.<br>**Result:** The dialog box is closed and FFB placement mode is activated. |
| 6 | Click the target position in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The selected block instance is inserted and the placement mode remains active so that more block instances can be inserted. To end insert mode press **Esc**. |

# Assign actual parameters

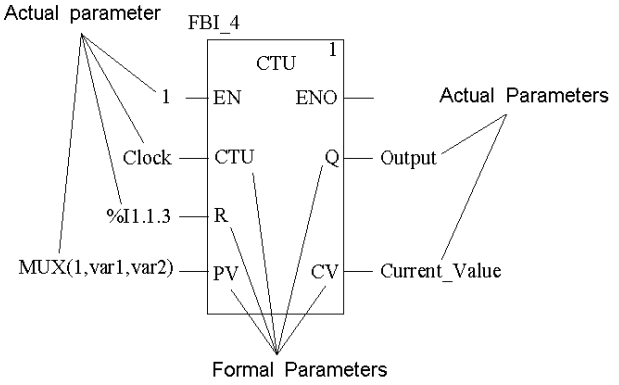
## Introduction

Inputs and outputs are required to transfer values to or from an FFB. These are called formal parameters.

Objects are linked to formal parameters; these objects contain the current process states. They are called actual parameters.

The data type of the actual parameter must match the data type of the input/output (formal parameter). The only exceptions are generic inputs/outputs whose data type is determined by the actual parameter. If all actual parameters consist of literals, a suitable data type is selected for the function block.

Formal and Actual Parameters:




Permitted actual parameters are:

- Variables
  - Enter already declared variables *(see page 643)*
  - Enter undeclared variables *(see page 645)*
- Formal parameters of other function blocks *(see page 647)*
- Literals *(see page 650)*
- ST Expression *(see page 651)*
  ST expressions as formal parameters on FBB inputs are an extension of IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by selecting the **Usage of ST expressions** checkbox.
- Links *(see page 690)* to other FBD objects

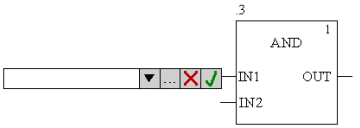
**NOTE:** Please take note of the section Information about Programming *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

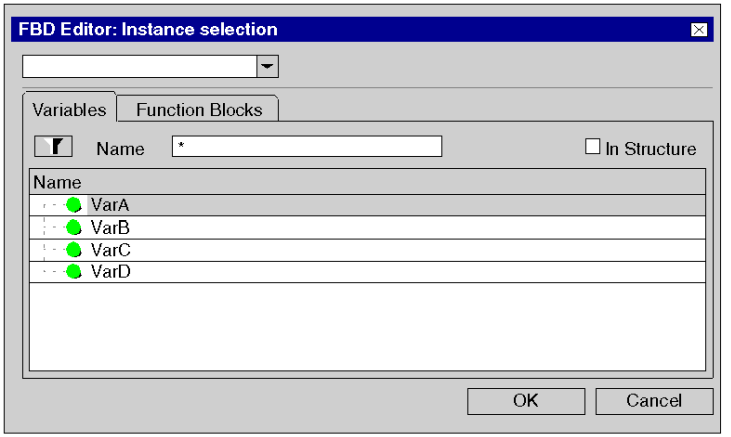
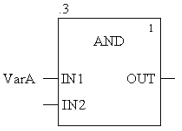
You can make actual parameter assignments for

- each pin individually
  or
- with the function input assistant *(see page 652)* for all pins

## Assign declared variables to a pin

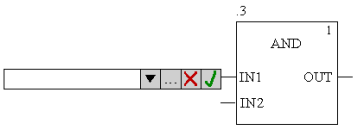
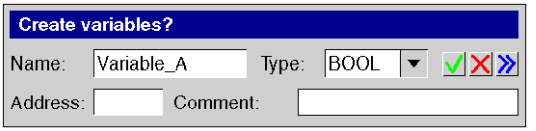
Process the following steps to assign undeclared variables to a pin. (See also *Creation of EDT type variable instances, page 370*.)

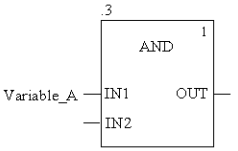
| Step | Action |
|---|---|
| 1 | Open a data selection box for the desired FFB pin by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection...** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |

| Step | Action |
|---|---|
| 2 | You have the following options to enter variables:<br>● You can now enter the variable name and confirm using **Enter**or the ✓button.<br>● You can select the variable names from the list of recently used names using the ▼ symbol and confirm using **Enter**or the ✓button.<br>● You can delete your entry by pressing **Esc**or clicking the ✕ button.<br>● Using the **button...** open a variable selection dialog *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example of a variable selection dialog box:<br>  <br><br>**Result:** The selected variable is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example:<br>  <br> |

## Assign undeclared variables to a pin

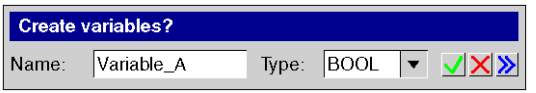
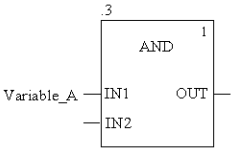
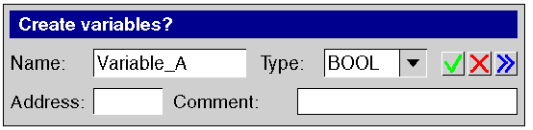
Process the following steps to assign undeclared variables to a pin:

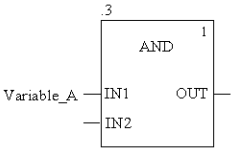
| Step | Action |
|---|---|
| 1 | Open a data selection box for the desired FFB pin by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection...** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the symbol.<br>**Result:**<br>A data selection box *(see page 1911)* opens.<br>  |
| 2 | Enter the variable name and confirm using **Enter**or the symbol.<br>**Result:** The variable declaration dialog box opens.<br>  |
| 3 | Change the default data type, if desired.<br>**Note:** If you wish to declare the variable later, close the dialog box using the symbol. In this case, the variable name is enabled on the pin, but not declared. |
| 4 | If you do want to assign the variable an address and/or comment, use the symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |

| Step | Action |
|---|---|
| 5 | Confirm with **Enter** or the  symbol.<br>**Result:** The variable is declared and enabled on the selected pin.<br>  |
| 6 | Enter all actual parameters in this way. |

## Declaring variables on a pin

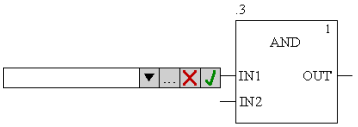
Process the following steps to declare variables on a pin:

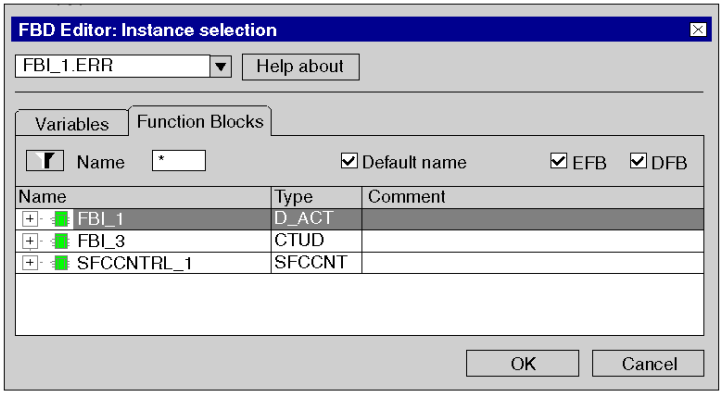
| Step | Action |
|---|---|
| 1 | Open the variable declaration dialog box for the desired variable by.<br>● Select an undeclared variable (red wavy line under the variable name) and the **Create Variable** command from the shortcut menu.<br>● Select an undeclared variable (red wavy line under the variable name) and the **Shift**+**Enter** key combination.<br>**Result:** The variable declaration dialog box opens.<br>  |
| 2 | If you do not want to assign an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears.<br>  |
| 3 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |

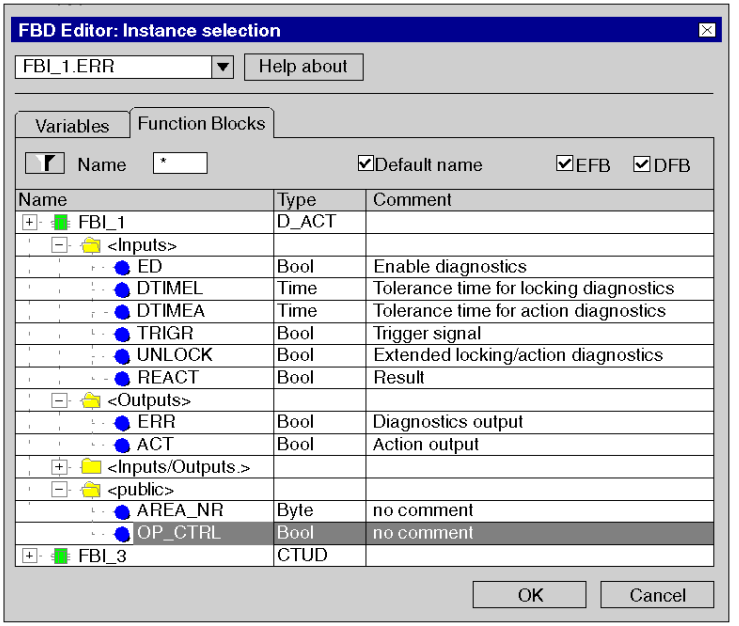
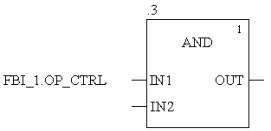
| Step | Action |
|---|---|
| 4 | Confirm with **Enter** or the ✓ symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears.<br>  |

## Assigning formal parameters to a pin

Process the following steps to assign formal parameters to a pin:

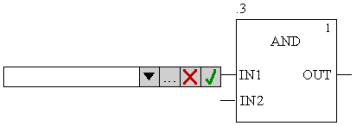
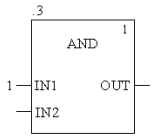
| Step | Action |
|---|---|
| 1 | Select the desired FFB pin. |
| 2 | Open a data selection box by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |
| 3 | You have the following options to enter the formal parameter:<br>● You can enter the formal parameter name and confirm using **Enter** or the ✓ button.<br>● You can select the formal parameter name from the list of recently used names using the ▼ symbol and confirm using **Enter**or the ✓button.<br>● You can delete your entry by pressing **Esc**or clicking the ✗ button.<br>● Using the **button...**to open a variable selection dialog box *(see page 1182)*. |

| Step | Action |
|---|---|
| 4 | Select the **Function Block** tab.<br>**Result:**<br>The function block instances are displayed.<br>  |

| Step | Action |
|---|---|
| 5 | Expand the display of the inputs, outputs and public variables by clicking their **+** symbols.<br>**Result:**<br>You can now see all available inputs, outputs and public variables of the function block.<br>  |
| 6 | Select the desired formal parameter and confirm with **OK**. |
| 7 | **Result:** The formal parameter selected is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example:<br>  |

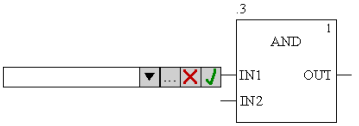
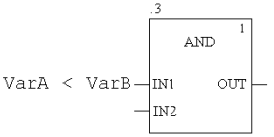
## Assigning literals to a pin

Process the following steps to assign literals to a pin:

| Step | Action |
|---|---|
| 1 | Select the desired FFB pin. |
| 2 | Open a data selection box by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |
| 3 | Enter the literal (e.g. `0`, `1`, `TRUE`, `FALSE`, `1.0`, `3.5`, `t#2ms`) and confirm using the **Enter**button.<br>**Tips:**When in the **Tools** → **Project Settings** dialog box, on the **Language extensions** tab, the **Allow leading digits** checkbox is cleared, the following simplifications for entering literals are possible:<br>● When assigning `TIME` literals to a pin of the `TIME` data type, it suffices to enter the numerical value and the unit (such as `h`, `m`, `s`). The prefix (`t#`) is automatically added.<br>● When assigning `REAL` literals to a pin of the `REAL` data type, for integers it suffices to enter the numerical value. The decimal marker (`t#`) is automatically added.<br>**Result:** The literal is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example:<br>  |

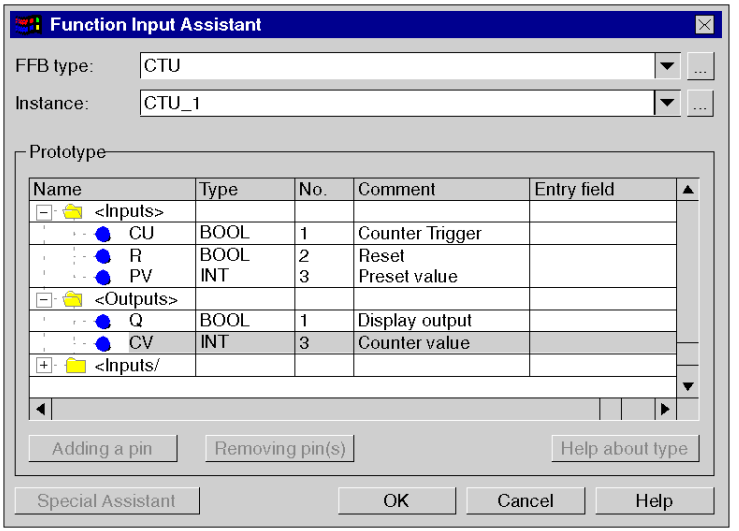
## Assigning ST expressions to a pin

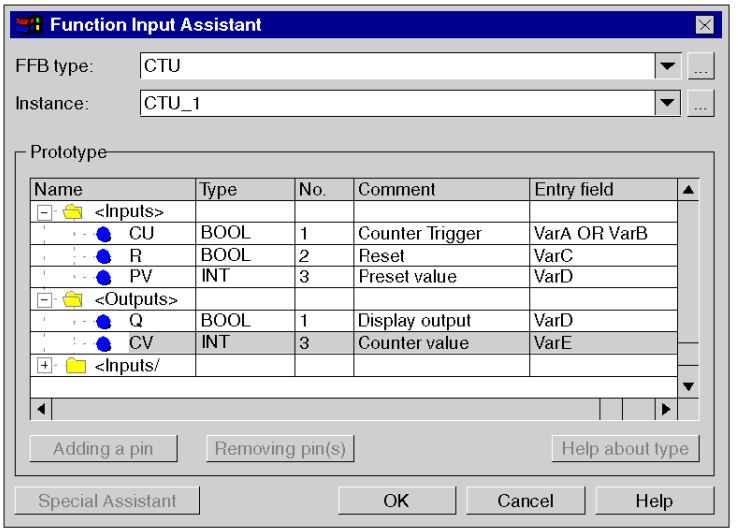
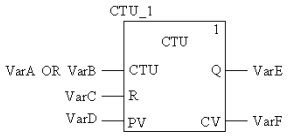
Process the following steps to assign ST expressions to a pin:

| Step | Action |
|---|---|
| 1 | Make sure that the **Usage of ST expressions** checkbox is activated in the **Tools** → **Project Settings** dialog box. |
| 2 | Select the desired FFB pin. |
| 3 | Open a data selection box by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |
| 4 | Enter the ST expression (e.g. `MUX(1,var1,var2)`, `VarA * VarB`, `VarA < VarB`) and confirm pressing **Enter**.<br>**Result:** The ST expression selected is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example:<br>  <br>If the variables have already been declared (see also *Creation of EDT type variable instances, page 370*), the procedure ends here.<br>If the variables have not yet been declared, continue with step 5. |
| 5 | Declare all variables used, either:<br>● using the data editor *(see page 370)*<br>or<br>● using **Create variable** in the context menu (possible when only one variable of the ST statement is undeclared). |

## Actual parameter assignment using the function input assistant

Process the following steps to change actual parameter assignment with the function input assistant:

| Step | Action |
|---|---|
| 1 | Select the desired FFB. |
| 2 | Open the function input assistant *(see page 1194)*:<br>● The **Edit → FFB Input Assistant... menu command**..<br>● The **FFB Input Assistant... menu command**from the shortcut menu.<br>or<br>● The **Ctrl+I** key combination.<br>**Result:**<br>The function input assistant will be opened.<br>  |

| Step | Action |
|---|---|
| 3 | Double click the **Entry Field** cell of the first formal parameter and enter the actual parameter to be used.<br>You have the following options to enter the variables/addresses:<br>● You can enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the **button...**to open a variable selection dialog box *(see page 1182)*.<br>Assign an actual parameter to all formal parameters of the function block in this way.<br>Example:<br>  |
| 4 | Confirm the entries using the **OK** button.<br>**Result:** The actual parameter is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example:<br>  |

# Using Public Variables

## Introduction

In addition to inputs/outputs, some function blocks also provide public variables.

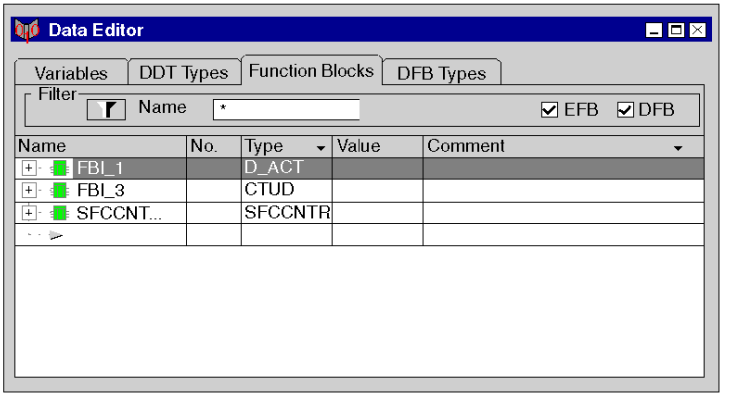
These variables transfer statistical values (values that are not influenced by the process) to the function block. They are used for setting parameters for the function block.

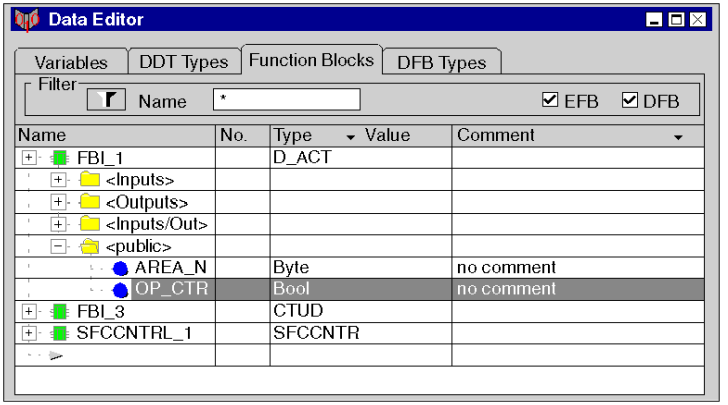
The assignment of values to public variables is made via their initial values.

Public variables are read via the instance name of the function block and the names of the public variables.

## Assigning Values to Public Variables

Carry out the following steps to assign a value to a public variable:

| Step | Action |
|---|---|
| 1 | Place a function block in the section, see also *Calling an FFB via the data selection, page 637*. |
| 2 | Open the data editor, see also *Data Editor Access, page 328*. |
| 3 | Select the **Function Block** tab.<br>**Result:**<br>The function block instances are displayed.<br>  |

| Step | Action |
|---|---|
| 4 | Expand the display of the public variables by clicking the respective **+** symbols.<br>**Result:**<br>You can now see all available public variables of the function block.<br>  |
| 5 | Enter the desired value for the public variable in the **Value** box. |

## Reading public variables

See *Assigning formal parameters to a pin, page 647*

# Expanding Functions

## Introduction

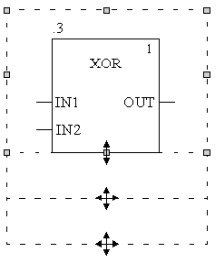
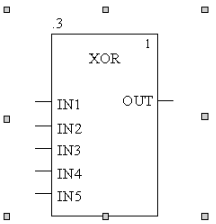
The number of inputs can be increased with some elementary functions.

You find out which functions can be expanded by referring to the descriptions for the individual functions.

**NOTE:** Only expand the function by the number of inputs actually required as non-occupied inputs are assigned a 0 as standard.

## Expanding a function

Carry out the following steps to expanding a function:

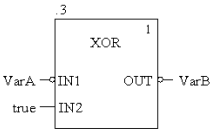
| Step | Action |
|---|---|
| 1 | Select the function. |
| 2 | Position the mouse pointer on the bottom point to be expanded. |
| 3 | Press the left mouse button and drag the function to the required size.<br>Example:<br>  |
| 4 | Release the mouse key.<br>**Result:** The desired number of inputs/outputs have been added.<br>Example:<br>  |

# Negating FFB Pins

## Introduction

FFB pins (inputs and outputs) of the data type `BOOL` can be negated.

Negated pins are identified with a circle symbol:




**NOTE:** Negated pins must be connected with a graphical link or a variable. Open negated pins are not allowed.

## Negating an FFB pin

FFB pins can be negated in the following ways:

- Select the pin to be negated and execute the menu command **Negate Pin** in the shortcut menu (right-click).
  or
- Launch the Inversion tool using
  - the menu command **New** → **Inversion Tool**,
  - the menu command **Inversion Tool** in the shortcut menu (right-click)
    or
  - the icon 

  and click on the pin to be negated.

# EN and ENO Show and Hide

## Introduction

With all FFBs, one EN input and one ENO output can be configured, see also EN and ENO *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

## EN and ENO show or hide

To show or hide EN and ENO, carry out the following steps:

| Step | Action |
|---|---|
| 1 | Select the FFB. |
| 2 | Open the properties dialog box *(see page 627)* for the FFB. |
| 3 | Check the **FFB Properties** check box **Show EN/ENO** to show EN/ENO, or uncheck it to hide EN/ENO.<br>**Note:** If EN or ENO is occupied by an actual parameter or connected with a link, EN/ENO cannot be hidden. |

# Modifying the execution order

## Introduction

The execution order is determined by the position of the FFBs in the section (executed from left to right and from top to bottom) and is displayed with execution numbers. If the FFBs are connected to a network using graphical links, the execution order is determined by the signal flow, see also section Execution order *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

You can already influence the execution order when creating the program.

- The use of links in place of actual parameters *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- The position of networks *(see Unity Pro, Program Languages and Structure, Reference Manual )*

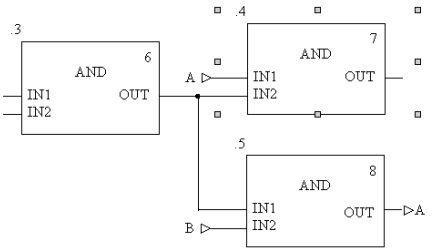
In some cases it might be necessary to explicitly modify the execution order.

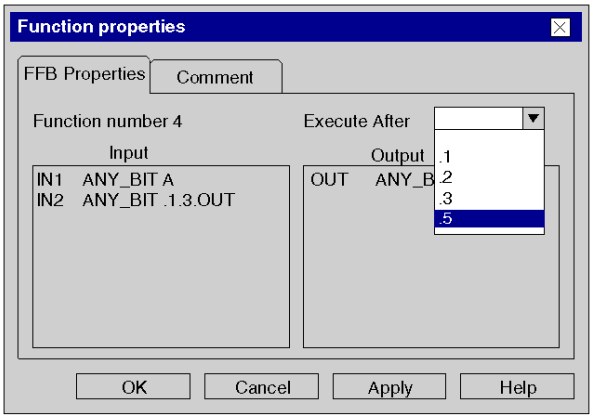
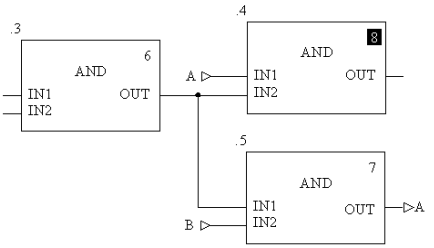
By using **Execute After** in an FFB properties dialog the execution order of two FFBs can be defined.

FFBs whose execution order has been explicitly modified receive the instance name / function number of the previously executed FFB as an additional entry in front of the execution number.

## Modifying the execution order

In the example, the execution order of FFB 1.4 and FFB 1.5 has to be switched. Process the following steps to modify the execution order using an FFB properties dialog box.

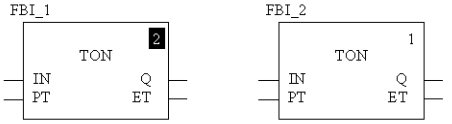
| Step | Action |
|---|---|
| 1 | Select the FFB to be executed second.<br>Example:<br>  |
| 2 | Open the properties dialog box *(see page 627)* for the FFB. |

| Step | Action |
|---|---|
| 3 | In the **Execute After** list box, select the function number / instance name of the FFB to be executed before the selected FFB.<br>Example:<br>  <br><br>**Result:**The execution sequence of both FFBs are exchanged. To indicate that the execution order was changed, the execution number is shown in a black field.<br>Example:<br>  <br><br>**Note:**Only one reference of an instance is allowed, e.g. the instance ".6" may only be referenced once. |

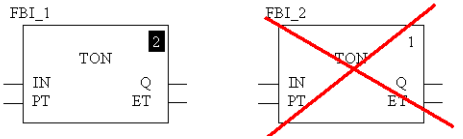
## Notes

When using **Execute after** please note the following:

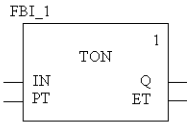
- If an FFB that is referenced by another FFB is deleted, the execution order attribute of the referencing FFB is reset.
  Example:
  **a.** FBI_1 has an "Execute After" reference to FBI_2.



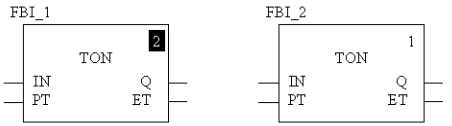

  **b.** FBI_2 is deleted.



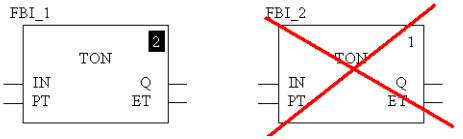

  **c.** For FBI_1 the attribute for determining the execution sequence is deleted.



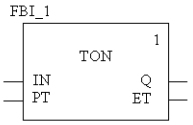

- If an FFB deletion operation carried out on a referenced FFB is undone using**Undo** it is reset to its former state.
  Example:
  **a.** FBI_1 has an "Execute After" reference to FBI_2.



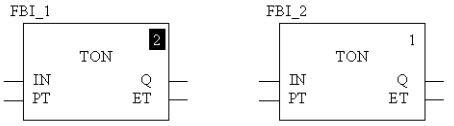

  **b.** FBI_2 is deleted.



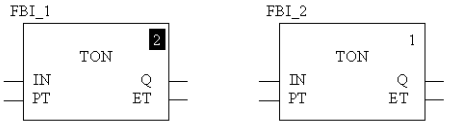
**c.** For FBI_1 the attribute for determining the execution sequence is deleted.



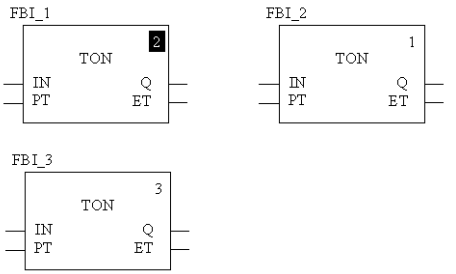

**d.** Execute from **Edit** → **Undo**.
**e.** FBI_2 is inserted again and FBI_1 and FBI_2 has an "Execute After" reference.




- If an FFB that has an "Execute After" reference pointing to another FFB is copied, the attribute which defines the execution order attribute of the FFB is reset during insertion.
  Example:
  **a.** FBI_1 has an "Execute After" reference to FBI_2.




**b.** FBI_1 is copied and the copy (FBI_3) inserted in the section.
**c.** For FBI_3 the attribute for determining the execution sequence is deleted.



# Replacing an FFB

## Access

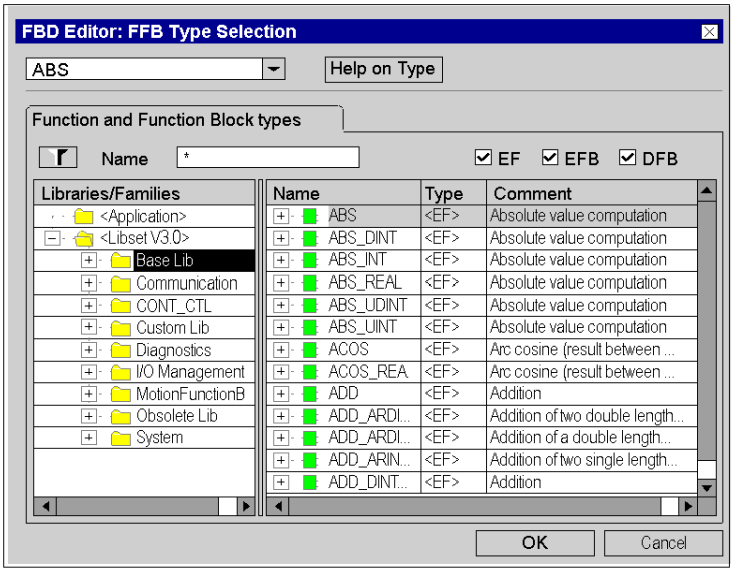
This is only available in FBD sections.

You have the following options to access the **Replace FFB** feature:

- Select the **Edit** → **Replace FFB** menu command.
- Select the **Replace FFB** command in the context menu (right-click on one or more FFBs in the section).

## Replace FFB

Selecting the **Replace FFB** feature opens the **FBD-Editor: FFB Type Selection** dialog box.




For more information about this dialog box see *FFB Selection Dialog box, page 1184*.

Select the desired type and confirm with **OK**. The FFBs selected in the FBD section are replaced with the new type.

Links and variables connected will be retained if possible (if a pin on the same position of the new type exists).

## Type Changing

There are different cases of type changing:

| Type Changing | Description |
|---|---|
| EF -> EF | changes the type for the existing EF only |
| EF -> FB | creates a new instance of the FB type |
| FB -> EF | creates the new EF template,<br>does not delete the old FB instance in the Data Editor (even if it is the only instance) |
| FB -> FB | creates the new FB instance,<br>does not delete the old FB instance in the Data Editor (even if it is the only instance) |

## Undo/Redo

**Replace FFB** supports Undo/Redo, which means that the original state will be recreated on Undo (of course without deleting the new FFBs from Data Editor) and the state after replacement on Redo.

## Online Mode

**Replace FFB** is supported in online mode.

# Properties dialog box for elementary functions, procedures and sub-routine blocks.

## Calling the properties dialog box

see *Displaying the properties, page 627*

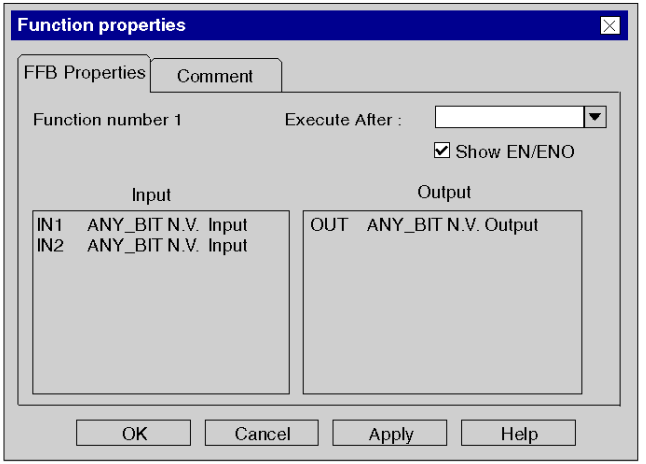
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **FFB Properties**
  General information about functions/procedures/sub-routine calls is displayed in this tab and you can change their order of execution.
- **Comment**
  You can enter a comment about the functions/procedures/sub-routine calls in this tab.

## FFB Properties tab

Representation of the **FFB Properties** tab:



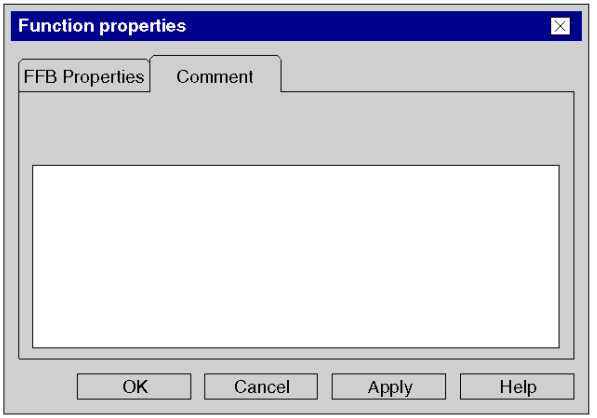

Elements of the **FFB Properties** tab:

| Element | Description |
|---|---|
| **Function number** | Specifies the name of the function number and the current execution order. This name cannot be modified. |
| **Execute After** | You can define the FFB execution order in this list. Select the FFB from this list after which the current FFB should be executed, see also *Modifying the execution order, page 659*. |

| Element | Description |
|---|---|
| **Show EN/ENO** | When you activate this checkbox the pins EN/ENO are displayed.<br>When you uncheck this checkbox the pins EN/ENO are no longer displayed.<br>**Note:** If EN or ENO is occupied by an actual parameter or connected with a link, the pins EN/ENO cannot be hidden. |
| **Input** | Displays the formal parameters, the data types and the actual parameter of the inputs.<br>If the input is not yet linked, the entry **N.L. appears**(not linked). |
| **Output** | Displays the formal parameters, the data types and the actual parameter of the outputs.<br>If the output is not yet linked, the entry **N.L. appears**(not linked). |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the**Comment**:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the FFB.<br>This comment is displayed in the tooltip when the cursor is placed over the function/procedure. |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

# Properties dialog box for elementary and derived function blocks (FBs)

## Calling the properties dialog box

see *Displaying the properties, page 627*

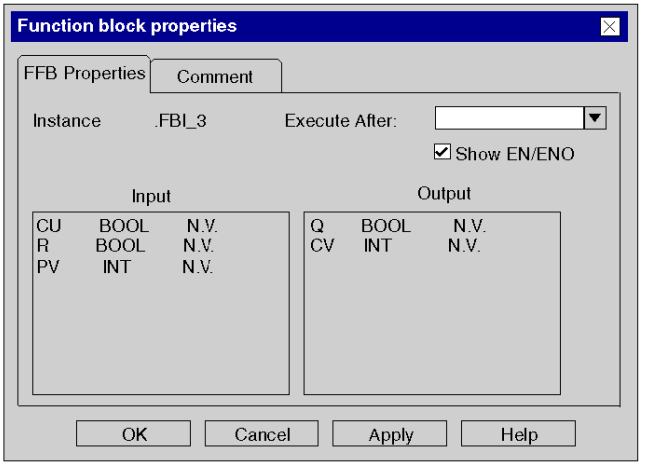
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **FFB Properties**
  General information about FBs is displayed in this tab and you can change their execution order.
- **Comment**
  A comment about the FB can be entered in this tab.

## FFB Properties tab

Representation of the **FFB Properties** tab:

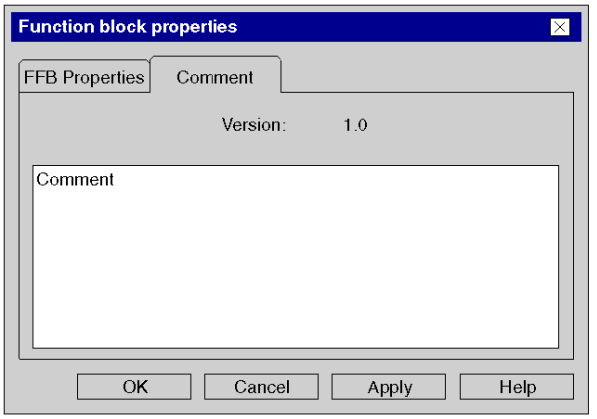


Elements of the **FFB Properties** tab:

| Element | Description |
|---|---|
| **Instance Name** | Specifies the name of the block instance. You can change this name in the data editor, see also *Managing of instances of data belonging to the family of function blocks (EF), page 359*. |
| **Execute After** | You can define the FFB execution order in this list. Select the FFB from this list after which the current FFB should be executed, see also *Modifying the execution order, page 659*. |
| **Show EN/ENO** | When you activate this checkbox the pins EN/ENO are displayed. When you uncheck this checkbox the pins EN/ENO are no longer displayed.<br>**Note:** If EN or ENO is occupied by an actual parameter or connected with a link, the pins EN/ENO cannot be hidden. |
| **Input** | Displays the formal parameters, the data types and the actual parameter of the FB inputs.<br>If the input is not yet linked, the entry **N.L. appears**(not linked). |
| **Output** | Displays the formal parameters, the data types and the actual parameter of the FB outputs.<br>If the output is not yet linked, the entry **N.L. appears**(not linked). |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the tab **Comment**:



Elements of the **Comment** tab:

| Element | Description |
|---|---|
| **Version:** | Specifies version of the FB code. |
| Text box | Enter a comment about the FB.<br>This comment is displayed in the tooltip when the cursor is placed over the FB. |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

# Section 21.3
## Returning from a Subroutine or DFB

### Overview

This section describes how to return from a subroutine or DFB to the FBD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Returning from a Subroutine or DFB | 671 |
| Return object properties dialog box | 673 |

# Returning from a Subroutine or DFB

## Introduction

Every subroutine and every DFB (derived function block) is exited after being processed, i.e. you return to the main program called.

If the subroutine/DFB is left prematurely, the return to the main program can be forced by the return object.

If the status of the left link is 1, a return from the subroutine or DFB to the main program is performed.

Return objects can only be used in subroutines or DFBs. They cannot be used in main programs.

To generate a conditional return, a return object is linked to a Boolean FFB output.

## Selecting the return object

You have the following options to select return objects:

- Use the **Edit** → **New** → **Return** menu command.
  or
- Choose the  symbol.

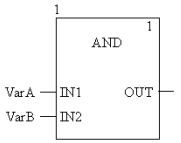
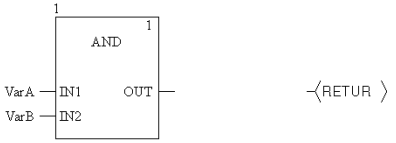
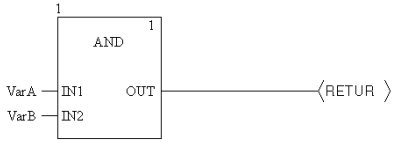
The active placement mode for return objects is indicated by the  cursor symbol.

## Note

In the **Tools** → **Options** dialog box, in the **Data and Languages** tab, if the **Automatically affect a variable on a new graphical object** check box is activated, its properties dialog box is automatically opened when an object is placed. The procedure explained here is valid for **deactivated** check boxes.

## Executing a return

Process the following steps to execute a return:

| Step | Action |
|---|---|
| 1 | Create a subroutine *(see page 481)* or a DFB *(see page 1255)* in the FBD programming language. |
| 2 | Create the logic for the subroutine/DFB. |
| 3 | Create the logic for return.<br>Example:<br>  |
| 4 | Select the return object. |
| 5 | Click the target position in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:**The return object is inserted.<br>Example:<br>  |
| 6 | Create a connection between the return logic and return object using a graphical link *(see page 693)*.<br>Example:<br>  |

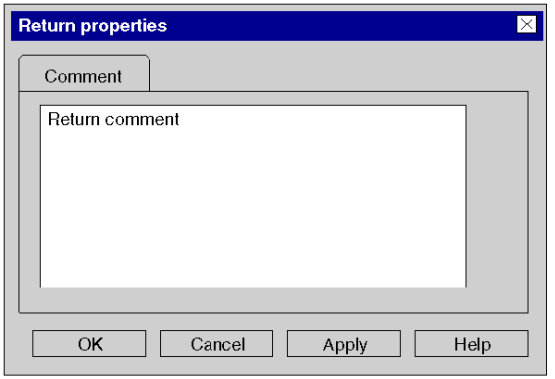
# Return object properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 627*

## Structure of the properties dialog box

Representation of the properties dialog box:




Elements of the properties dialog box:

| Element | Description |
|---|---|
| Text box | Enter a comment about the return object. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 21.4
# Calling a Subroutine

## Calling a subroutine

### Introduction

In FBD, subroutines are called using a special subroutine block, see also section Subroutine call *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

The subroutine to be called must be located in the same task as the FBD section calling.

For an unconditional subroutine call *(see page 675)*, EN and ENO of the subroutine block are hidden *(see page 658)* or the input EN is assigned the value 1.

For a conditional subroutine call *(see page 676)*, EN and ENO of the subroutine block are shown *(see page 658)* and the input EN is linked with a Boolean FFB output.

For a parallel call of several subroutines *(see page 677)*, EN and ENO of the subroutine blocks are shown *(see page 658)* and the output ENO is connected to the EN input of the next subroutine block.

Subroutine calls are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow subroutines** check box.

### Activating the placement mode for subroutine blocks

You have the following options to activate the placement mode for subroutine blocks:

- Use the **Edit** → **New** → **Subroutine** menu command.
- Use the **Subroutine** menu command from the shortcut menu.
  or
- Select the  symbol.

The active placement mode for subroutine blocks is indicated by the  cursor symbol.

### Note

In the **Tools** → **Options** dialog box, in the **Data and Languages** tab, if the **Automatically affect a variable on a new graphical object** check box is activated, its properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the check box is **cleared**.

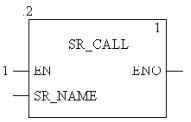
## Placing subroutine blocks:

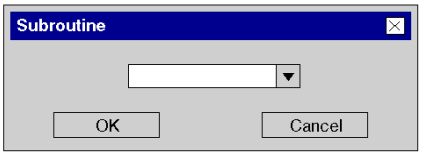
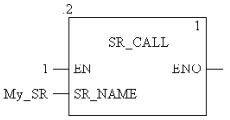
Placing subroutine blocks:

| Step | Action |
|---|---|
| 1 | Activate the placement mode for subroutine blocks, see also *Activating the placement mode for subroutine blocks*, *page 674*. |
| 2 | Click the target cell in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The selected subroutine block is inserted and the selection mode is active again. |
| 3 | • Placing other subroutine blocks:<br>Click the target cell in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>• To return to selection mode:<br>Press the **Esc** key.<br>• To insert other objects:<br>Select the object you wish to insert. |

## Unconditional subroutine call

Unconditional subroutine call:

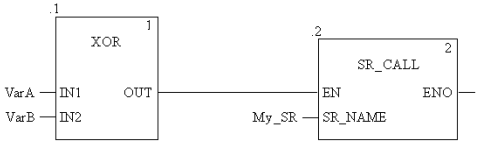
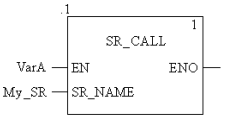
| Step | Action |
|---|---|
| 1 | Place a subroutine block in the section, see also *Placing subroutine blocks:*, *page 675*. |
| 2 | Place the mouse pointer on the subroutine block.<br>**Result:** The mouse pointer changes its symbol to  . |
| 3 | Double-click the `EN` input of the subroutine block.<br>**Result**: A data selection box is opened. |
| 4 | Enter `1` or `TRUE` for the unconditional subroutine call.<br>Example:<br>  |

| Step | Action |
|---|---|
| 5 | Double-click the SR_NAME input of the subroutine block.<br>**Result**: A selection dialog box for selecting the subroutine to be called is opened.<br>Example:<br>  |
| 6 | Enter the name of the subroutine, or click the ▼ symbol and select the subroutine to be called from the list. Confirm the entry with **Enter**.<br>**Result:** The subroutine call is inserted into the section and a syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  <br>If the subroutine has already been created (see also *Create a Sub-Program Section (SR), page 481*), the procedure ends here.<br>If the subroutine has not yet been created, continue with step 7. |
| 7 | Create the subroutine, see also*Create a Sub-Program Section (SR), page 481*. |

## Conditional subroutine call

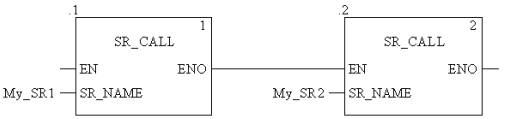
Conditional subroutine call:

| Step | Action |
|---|---|
| 1 | Execute the *Unconditional subroutine call, page 675* procedure, but do not assign a value to the EN input. |
| 2 | Create a Boolean logic for the subroutine call. |

| Step | Action |
|---|---|
| 3 | Connect the logic's Boolean output to the EN input of the subroutine block.<br>Example:<br>  <br><br>**Note**: To execute the conditional subroutine call you can also allocate any Boolean variable to the EN input.<br>Example:<br>  <br> |

## Parallel call of several subroutines

Calling several subroutines:

| Step | Action |
|---|---|
| 1 | Execute the *Unconditional subroutine call, page 675* procedure for all subroutines to be called in parallel. |
| 2 | Connect the ENO outputs and the EN inputs of the subroutine block.<br>Example:<br>  <br> |
| 3 | Possibly create a Boolean logic for the first subroutine block. |

# Section 21.5
## Jumps within the Current Section

### Overview

This section describes jumps within the current FBD section.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Jump within the current section | 679 |
| Jump object properties dialog box | 683 |

# Jump within the current section

## Introduction

Jump objects execute jumps to a jump target *(see page 686)* within the current section.

When the status of the left link is 1, a jump is made to a label (in the current section).

To generate a conditional jump, a jump object is linked to a Boolean FFB output.

To generate an unconditional jump, the jump object is assigned the value 1 using the `AND` function for example.

## Selecting the jump object

You have the following options to select jump objects:

- Use the **Edit** → **New** → **Jump** menu command.
  or
- Choose the  symbol.

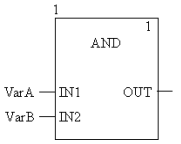
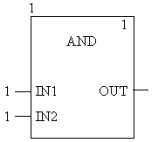
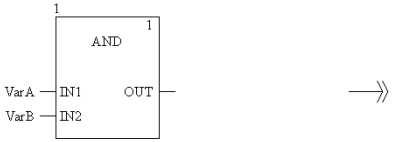
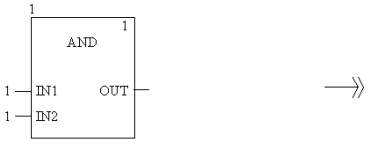
The active placement mode for jump objects is indicated by the  cursor symbol.

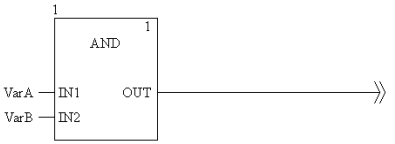
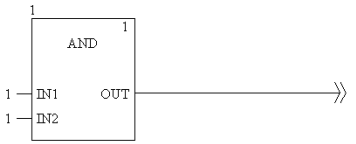
## Note

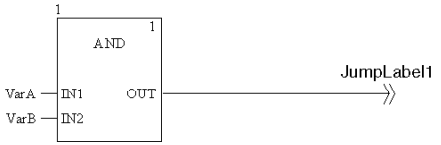
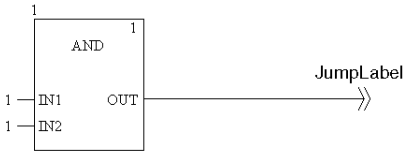
In the **Tools** → **Options** dialog box, in the **Data and Languages** tab, if the **Automatically affect a variable on a new graphical object** check box is activated, its properties dialog box is automatically opened when an object is placed. The procedure explained here is valid for **deactivated** check boxes.

## Executing a jump

Process the following steps to execute a jump:

| Step | Action |
|---|---|
| 1 | Create the logic for the jump.<br>Example: Logic for conditional jump:<br>  <br><br>Example: Logic for unconditional jump:<br>  <br> |
| 2 | Select the jump object. |
| 3 | Click the target position in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The jump object is inserted.<br>Example: Logic for conditional jump:<br>  <br><br>Example: Logic for unconditional jump:<br>  <br> |

| Step | Action |
|---|---|
| 4 | Create a link between the jump logic and jump object using a graphical link *(see page 693)*.<br>Example: Logic for conditional return:<br>  <br><br>Example: Logic for unconditional return:<br>  <br> |
| 5 | To enter the jump target, open the properties dialog box *(see page 627)* of the jump object.<br>**Result**: The properties dialog box for the jump object is opened, see also *Jump object properties dialog box, page 683*. |

| Step | Action |
|---|---|
| 6 | You have the following options to enter the jump target:<br>● You can now enter the name of the jump target and confirm using **Enter**.<br>or<br>● You can select the name of the jump target from the list of available jump targets using the ▾ symbol and confirm using **Enter**.<br>**Result:** The jump target is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example of a conditional jump:<br>  <br><br>Example of an unconditional jump:<br>  <br><br>If the jump target has already been defined (see also Jump target *(see page 685)*), the procedure ends here.<br>If the jump target has not yet been declared, continue with step 7. |
| 7 | Specify the jump target, see also Jump target *(see page 685)*. |

# Jump object properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 627*

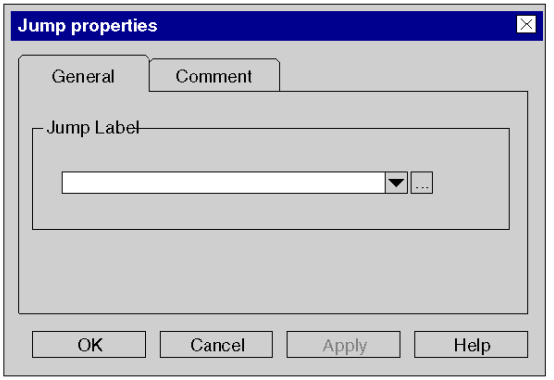
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **General**
  You can enter the jump target of the jump *(see page 680)* in this tab.
- **Comment**
  A comment about the jump object can be entered in this tab.

## General tab

Representation of the **General** tab:

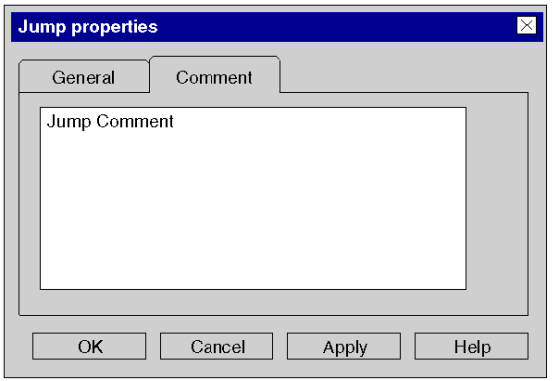



Elements of the **General** tab:

| Element | Description |
|---|---|
| **Jump Label** | Enter the jump target in this text box, see also *Definitions of Jump Labels, page 686*. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the jump object. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 21.6
## Definition of the Jump Target (Jump Labels)

### Overview

This section describes how to define jump targets in the FBD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Definitions of Jump Labels | 686 |
| Jump label objects properties dialog box | 688 |

# Definitions of Jump Labels

## Introduction

Jump labels create a jump target for jumps *(see page 679)* within the current section.

Jump labels are represented as text with a colon at the end.

The text is limited to 32 characters and must be unique across the entire section. The text must conform to general naming conventions.

Jump labels can only be placed between the first two grid points on the left edge of the section.

**NOTE:** Jump labels may not "dissect" networks, i.e. an assumed line from the jump label to the right edge of the section may not be crossed by any object. This also applies to the connections.

To find a specific jump label use the **Go to** dialog box.

## Selecting the jump label

You have the following options to select jump labels:

- Use the **Edit** → **New** → **Jump Label** menu command.
  or
- Choose the JL symbol.

The active placement mode for jump labels is indicated by the JL cursor symbol.

## Note

In the **Tools** → **Options** dialog box, in the **Data and Languages** tab, if the **Automatically affect a variable on a new graphical object** checkbox is activated, its properties dialog box is automatically opened when an object is placed. The procedure explained here is valid for **deactivated** checkboxes.

## Placing a jump label

Placing a jump label:

| Step | Action |
|---|---|
| 1 | Select the jump label. |
| 2 | Click with the mouse on the target position between the first two grid points on the left edge of the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target positions between the first two grid points on the left edge of the FBD section, and press **Enter**.<br>**Result:** The jump label is inserted and the selection mode is active again.<br>Example:<br>`???:` |

| Step | Action |
|---|---|
| 3 | To insert further jump objects:<br>● Click the target cell in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Definition of the Name

Definition of the Name:

| Step | Action |
|---|---|
| 1 | To enter the name, open the properties dialog box *(see page 627)* for the jump label.<br>**Result**: The jump label properties dialog box is opened, see also *Jump label objects properties dialog box, page 688*. |
| 2 | You have the following options to enter the name of the jump label:<br>● You can now enter the name and confirm using **Enter**.<br>or<br>● You can select the name of the jump from the list of available jumps using the ▾ symbol and confirm using **Enter**.<br>**Result:** The name is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example:<br>`JumpLabel1:` |

# Jump label objects properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 627*

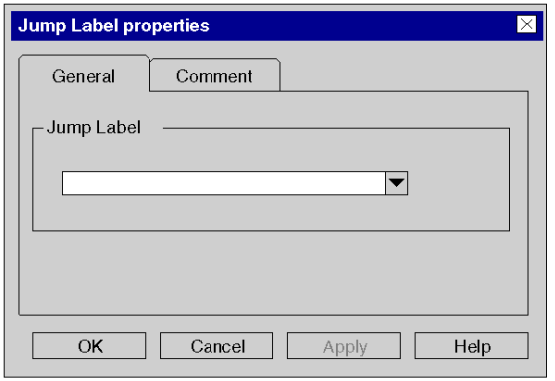
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **General**
  You can enter the name of the jump label *(see page 686)* in this tab.
- **Comment**
  A comment about the jump label can be entered in this tab.

## General tab

Representation of the **General** tab:

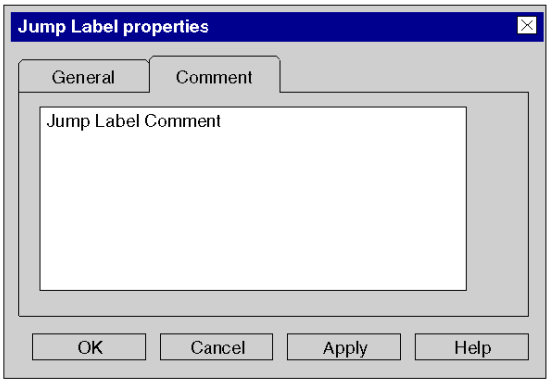



Elements of the **General** tab:

| Element | Description |
|---|---|
| **Jump Label** | You can enter the name of the jump label in this list box.<br>The text is limited to 32 characters and must be unique across the entire section. The text must conform to general naming conventions. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the jump label object. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 21.7
# Editing Links

## Overview

Editing links in the FBD programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Links | 691 |
| Placing a Link | 693 |
| Editing Links | 696 |

# Links

## Introduction

Links are vertical and horizontal connections between FFBs.

Attention should be paid to the following programming information:

- Links can be used for every data type.
- The data types of the inputs/outputs to be linked must be the same.
- Several links can be connected with one FFB output. However, only one may have an FFB output.
- Only inputs and outputs can be linked together. A link between several outputs cannot be made. This means that OR links are not possible using links in FBD. An `OR` function should always be used.
- An overlap of links with other objects is permitted.
- No loop can be configured with links because in this case the execution sequence in the section cannot be clearly determined. Loops must be terminated with actual parameters (see Terminating loops *(see Unity Pro, Program Languages and Structure, Reference Manual )*).
- To avoid links crossing each other, links can also be represented in the form of connectors *(see page 588)*.

**NOTE:** A detailed description about Links *(see Unity Pro, Program Languages and Structure, Reference Manual )* can be found in the Reference manual.

## Selecting links

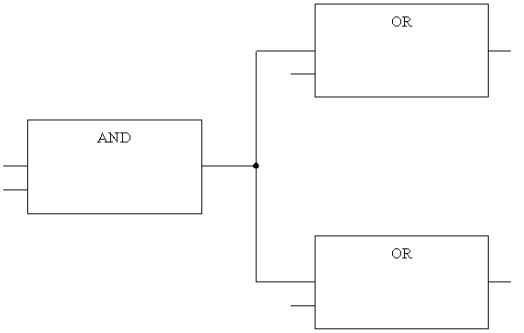
You have the following options to activate the placement mode:

- Use the **Edit** → **New** → **Link** menu command.
- Press the **F6** key.
  or
- Select the  symbol.

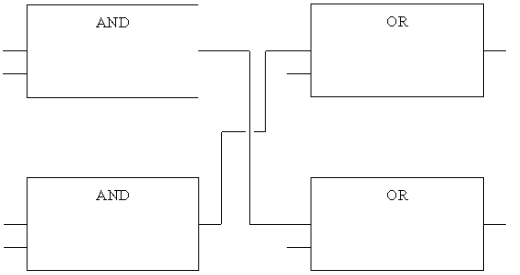
The + and  cursor symbols indicate which mode is active.

## Representation

The link points are identified by a filled-in circle.



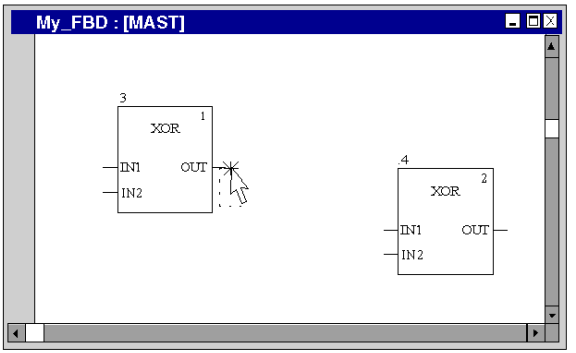
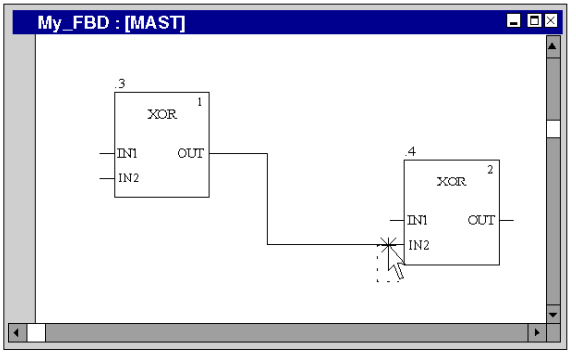

Crossed links are indicated by a "broken" link.

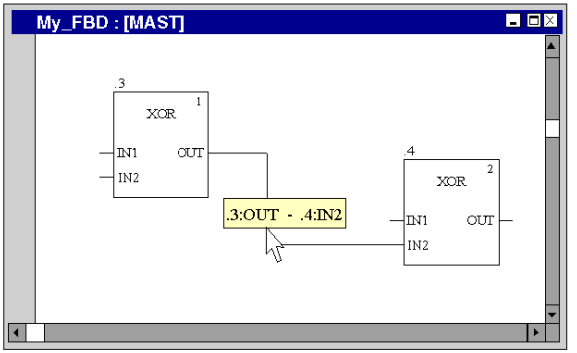


# Placing a Link

## Placing a link using the mouse

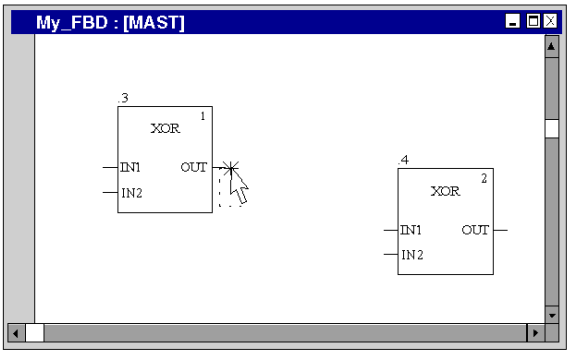
Carry out the following steps to place a link using the mouse:

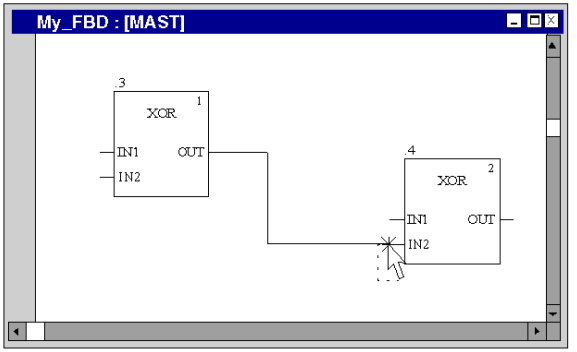
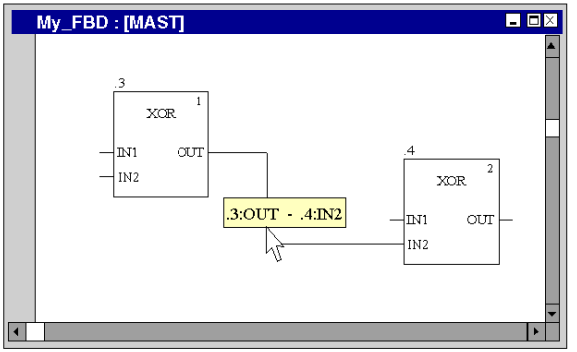
| Step | Action |
|---|---|
| 1 | Activate the placement mode for links, see also *Links*, *page 691*. |
| 2 | Place the mouse pointer on the start point of the link.<br>**Note:** The correct position is shown by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Click the left mouse button on the start point and drag mouse pointer to the target point of the link.<br>Example:<br>  <br>**Tips:**<br>● By clicking between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● You can leave this mode at any time by pressing **Esc**. |

| Step | Action |
|---|---|
| 4 | Click on the target position with the left mouse button.<br>**Result:** The link is inserted. When the cursor is placed over a link, the Quickinfo (Tooltip) shows the source and destination of the link.<br>Example:<br>  <br>See also *Editing Links, page 696* |

## Placing a link using the keyboard

Carry out the following steps to place a link using the keyboard:

| Step | Action |
|---|---|
| 1 | Activate the placement mode for links, see also *Links, page 691*. |
| 2 | Use the **cursor keys** to move the cursor to the start point of the link, and press **Enter**.<br>**Note:** The correct position is shown by the cursor symbol.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Use the **cursor keys** to move the cursor to the target position of the link.<br>Example:<br>  <br><br>**Tips:**<br>● By pressing the **Enter** key between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● You can leave this mode at any time by pressing **Esc**. |
| 4 | Press **Enter**.<br>**Result:** The link is inserted. When the cursor is placed over a link, the Quickinfo (Tooltip) shows the source and destination of the link.<br>Example:<br>  <br><br>See also *Editing Links, page 696* |

# Editing Links

## Editing links

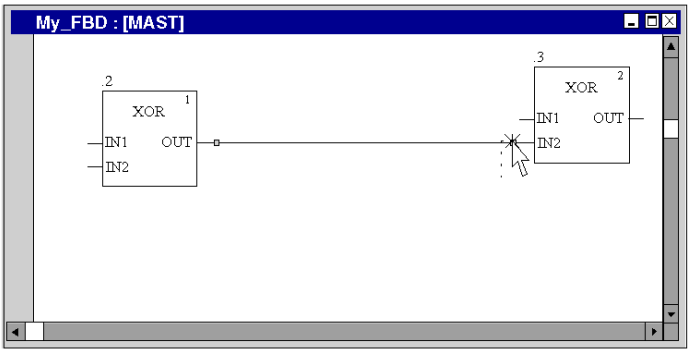
Links consist of one or more segments connected to one another.

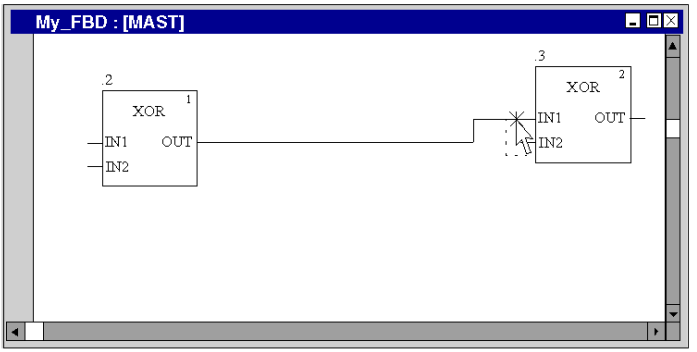
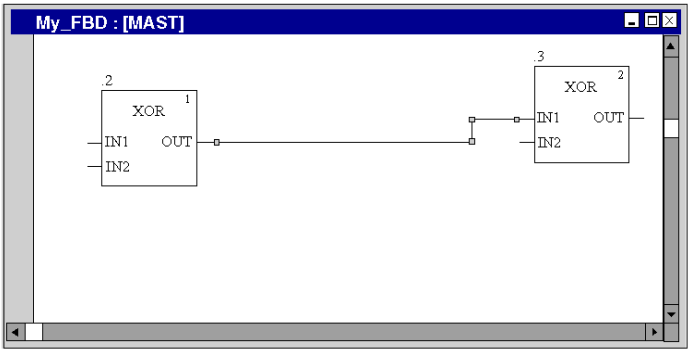
When cutting *(see page 621)*, deleting *(see page 621)*, copying *(see page 622)* and moving *(see page 623)* links, the action is executed for the entire link.

If a link is selected, the target points are displayed at their beginnings, ends and at every direction change

## Modifying connections

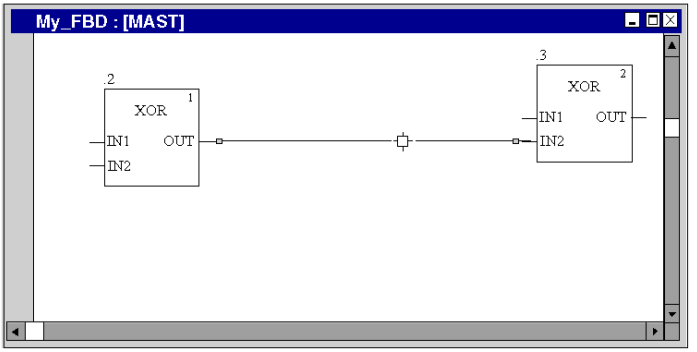
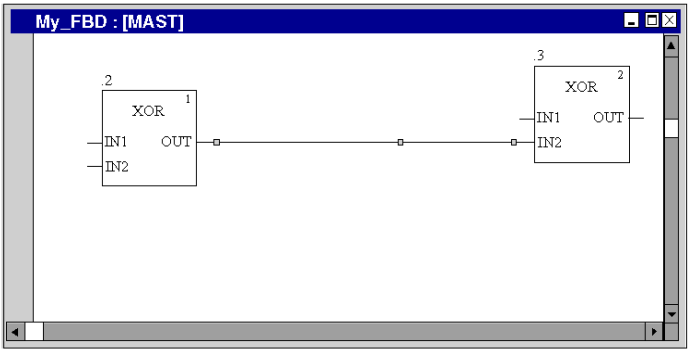
Modifying connections:

| Step | Action |
|---|---|
| 1 | Select the link and place the mouse pointer on the connection to be modified.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | Left-click the connection to be modified and drag the mouse pointer to the new connection.<br>Example:<br>  |
| 3 | Left-click the new target position.<br>**Result:** The link route is modified.<br>Example:<br>  |

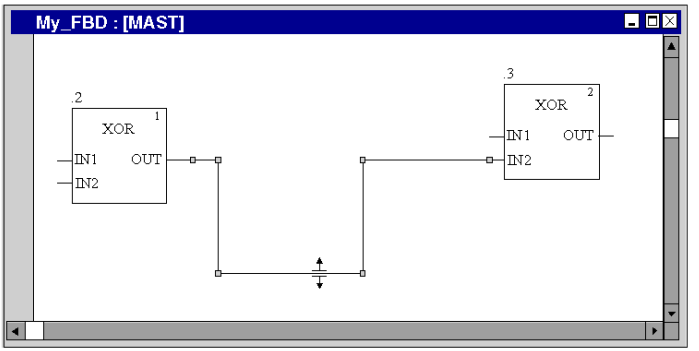
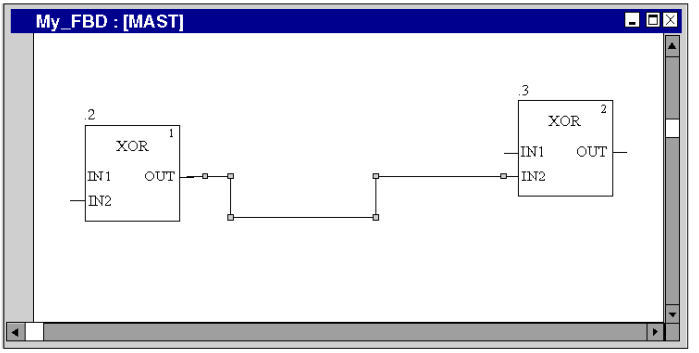
## Creating sizing handles

The creation of angles in links can only be made using sizing handles. If no sizing handles are available they can be created. Process the following steps to create sizing handles in links:

| Step | Action |
|---|---|
| 1 | Select the link. |
| 2 | Press the **Ctrl** key and place the mouse pointer at the target position.<br>**Note:** The correct position is shown by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Left-click the target position.<br>**Result:**<br>The sizing handle is created.<br>  |

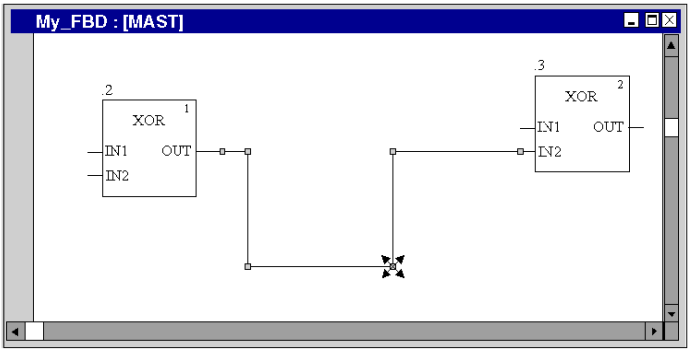
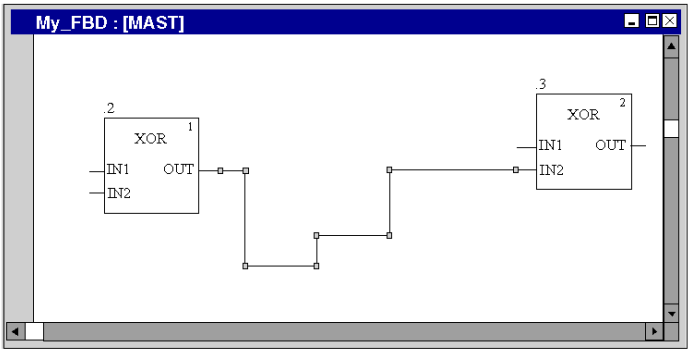
## Moving links vertically/horizontally

Process the following procedures to move a link segment vertically or horizontally:

| Step | Action |
|---|---|
| 1 | Select the link. |
| 2 | Place the mouse pointer on the item to be moved.<br>**Note:** The correct position is shown by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Click the left mouse button and drag the segment to the target position. |
| 4 | Release the mouse button.<br>**Result:**<br>The change is made.<br>  |

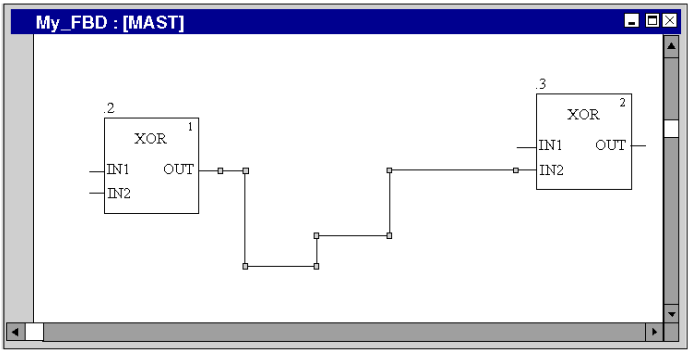
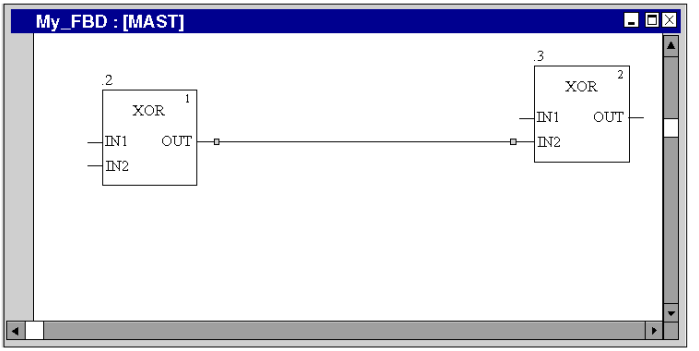
## Creating angles in FFB links

Process the following steps to create angles in links:

| Step | Action |
|---|---|
| 1 | Select the link. |
| 2 | Place the mouse pointer on a source position.<br>**Note:** The correct position is shown by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Click the left mouse button and drag the source point to the target position. |
| 4 | Release the mouse button.<br>**Result:**<br>The change is made.<br>  |

## Determining the optimum connection route

Determining the optimum link route:

| Step | Action |
|---|---|
| 1 | Select the link.<br>Example:<br>  |
| 2 | Use the **Edit** → **Link** → **Route** menu command.<br>or<br>Use the **Route** menu command from the shortcut menu.<br>**Result:** The current link route is changed to the optimum route.<br>Example:<br>  |

# Section 21.8
# Entering Comments

## Entering comments

### Introduction

Comments can be placed in FBD in the form of text objects.

Text objects can overlap other objects.

The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill additional grid units.

The same rules apply for entering text and navigating within text objects as for editing ASCII text in standard text editors. (To create a line break in text objects you must press the **Ctrl**+**Enter** key combination).

**NOTE:** Keep in mind that each change to a comment (e.g. change to a comment text, change to the size of the text object) makes it necessary to recreate the section involved (**Generation** → **Generate project**).

### Selecting the text object

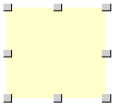
You have the following options to select text objects:

- Use the **Edit** → **New** → **Comment** menu command.
- Use the **Comment** menu command from the shortcut menu.
- Press the **F8** key.
  or
- Select the  symbol.

The active placement mode for text objects is indicated by the  cursor symbol.

## Placing text objects:

Placing text objects:

| Step | Action |
|---|---|
| 1 | Select the text object. |
| 2 | Click the target position in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The selected text object is inserted and the selection mode is active again.<br>Example:<br>  |
| 3 | Enter the comment. |
| 4 | Confirm the text entered by:<br>● Clicking with the mouse outside of the text object<br>or<br>● Pressing the **Enter** key. |
| 5 | To insert additional text objects:<br>● Click the target position in the FBD section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Section 21.9
# Online functions

## Online functions

### Online functions

The description of online functions is found in *Debugging in Function Block Diagram (FBD) Language*, *page 1440*.

# Section 21.10
## FBD Reference Data Type

## Reference Data Type in FBD

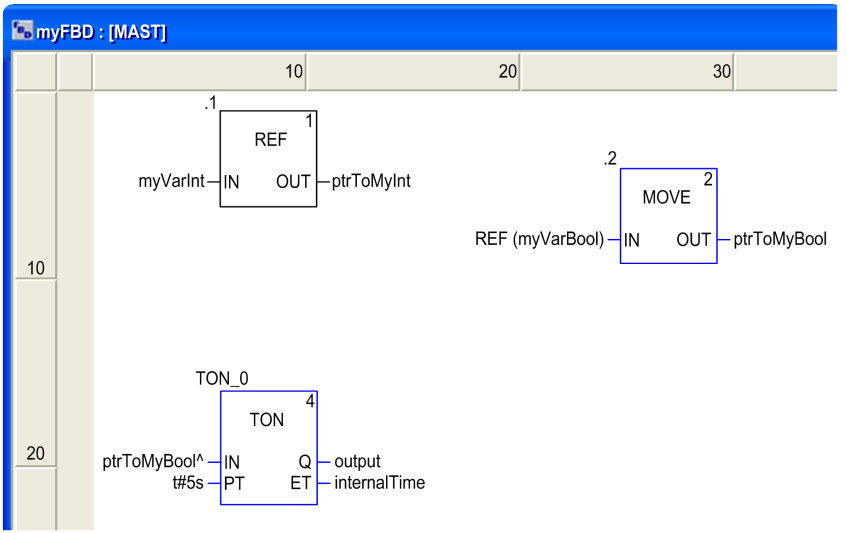
### Introduction

References to variables can be assigned in FBD application programs.

A reference can be de-referenced in FBD application programs.

For more information, refer to Reference Data Type *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

FBD Examples using the Reference Data Type:




The EF, `REF`, takes a variable as an input parameter (type `ANY`) and provides its address as output parameter (type `REF_ANY`).

The EF, `MOVE`, can assign a reference to a variable to another reference. If the input parameter is (`REF(MyVariable)`), `MOVE` has the same effect as the `REF` EF.

The EF, `TON`, takes `ptrToMyBool^` as an input parameter to dereference the reference ptrToMyBool.

# Section 21.11
# Export/Import

## Export/Import FBD Sections

### Export/Import

The description for exporting/importing sections is found in the chapter *Import / Export*, *page 1719*.

# Section 21.12
# Customize FBD Editor Colors

## Customize FBD Editor Colors

### Overview

To customize FBD editor colors, modify the FBDColors.ini in the Unity Pro folder.

For more detail on destination folder when Unity Pro is installed, refer to the *Unity Pro Installation Manual (see Unity Pro, Installation Manual)*.

**NOTE:** Make a copy of this file before modifying it.

The FBD editor colors are described in the table below.

### Description

This table describes the elements of the color file:

| Parameter | Default value | Description |
|---|---|---|
| `BkAnimation` | `192,192,192` | Editor background color when animation is ON and you are connected in programming mode. |
| `StatementError` | `255,0,0` | Element in error is underlined with this color. |
| `InspectBk` | `0,255,255` | Inspect window background color |
| `InspectText` | `0,0,0` | Inspect window text color |
| `InspectBkMin` | `255,255,0` | Inspect window background color when the value is less than the minimum value defined in the inspect window settings. |
| `InspectBkMax` | `255,0,255` | Inspect window background color when the value is greater than the maximum value defined in the inspect window settings. |
| `BkAnimMonitoring` | `255,211,211` | Editor background color when animation is ON and you are connected in monitoring mode. |
| `AnalyzeError` | `0,0,255` | Drawing color when an element generates an analyze error. |
| `GraphAnimBoolTrue` | `0,150,0` | Color of `ANY_BOOL` variable when its value is TRUE. |
| `GraphAnimBoolFalse` | `255,0,0` | Color of `ANY_BOOL` variable when its value is FALSE. |
| `BkComment` | `255,255,204` | Background color of comment area |
| `BkCommentEdit` | `225,225,225` | Background color of comment area when you are editing it. |

| Parameter | Default value | Description |
|---|---|---|
| BkAnimNumeric | 255,255,0 | Background color of numeric variables |
| BlockBreakpoint | 128,0,0 | Bullet color in the element where the breakpoint is set. |
| BlockCurrentStep | 255,255,0 | Color of the current element in debug mode |
| AnchorLink | 128,0,0 | Color of anchor link |

# Chapter 22
# LD Editor

## Overview

This chapter describes the menus and dialog boxes of the LD editor.

For a description of the syntax for the LD programming language, please refer to Ladder Diagram LD *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 22.1 | Creating a program with LD programming language | 710 |
| 22.2 | Editing Contacts | 738 |
| 22.3 | Editing Coils | 748 |
| 22.4 | Editing compare blocks | 761 |
| 22.5 | Editing operation blocks | 768 |
| 22.6 | Editing FFBs | 776 |
| 22.7 | Returning from a Subroutine or DFB | 811 |
| 22.8 | Jumps within the Current Section | 815 |
| 22.9 | Definition of the Jump Target (Jump Labels) | 821 |
| 22.10 | Editing Links | 827 |
| 22.11 | Entering Comments | 853 |
| 22.12 | Online functions | 855 |
| 22.13 | LD Reference Data Type | 856 |
| 22.14 | Export/Import | 857 |
| 22.15 | Customize LD Editor Colors | 858 |

# Section 22.1
## Creating a program with LD programming language

### Overview

This section describes the basics for creating a program in the LD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Structure of an LD program (ladder diagram) | 711 |
| Creating an LD program | 713 |
| Syntax and Semantics Check during programming | 717 |
| View for variables | 719 |
| Navigating with the Keyboard | 724 |
| Selecting objects | 727 |
| Deleting, cutting, copying, pasting and moving objects | 730 |
| Displaying the properties | 734 |
| Refining DFBs and subroutines | 735 |
| Inserting and Deleting Rows | 737 |

# Structure of an LD program (ladder diagram)

## Introduction

The structure of an LD program corresponds to a rung for relay switching.

The left power rail is located on the left side of the LD editor. This left power rail corresponds to the phase (L conductor) of a rung. With LD programming, in the same way as in a rung, only the LD objects which are linked to a power supply, that is to say connected with the left power rail, are "processed". The right power rail corresponds to the neutral conductor.

All contacts and FFB inputs must be directly or indirectly connected with the left power rail and all coils and FFB outputs must be directly or indirectly connected with the right power rail to allow current flow.

A group of connected objects that are not connected to other objects (except for the power rails), is called a network or rung.

## Properties of an LD program

Properties of an LD program:

- An LD section contains 11-64 columns and 17-2000 lines.
  The number of columns is set via the **Tools** → **Project Settings** dialog box, in the **Editors** tab in the **Number of columns** text box.
  **Note:**This setting is only valid for new sections and not for existing sections.
  The number of lines can be increased *(see page 737)* or reduced *(see page 737)* during the creation of section contents.
- LD programs are cell oriented, i.e. only one object can be placed in each cell.
- The processing sequence of the individual objects in an LD program is determined by the data flow within the section. Networks connected to the left power rail are processed from top to bottom (connection to the left power rail). Separate networks within the section are processed in the order in which they are placed (top to bottom) (see also section Execution order and signal flow *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual).
- A syntax and semantics check *(see page 717)* is performed immediately after the statement is entered. The result of this check is displayed in colored text and objects.
- Syntactically or semantically incorrect sections can be saved.

## LD program objects

The objects of the LD programming language help to divide a section into a number of:
- Contacts *(see page 738)*,
- Coils *(see page 748)*,
- EFs and EFBs *(see page 776)* (Elementary Functions and Elementary Function Blocks),
- DFBs *(see page 776)* (derived function blocks)
- Procedures *(see page 776)*
- operate blocks *(see page 768)*,
- compare blocks *(see page 761)*,
- jumps *(see page 815)* within the section and
- subroutine calls *(see page 757)*

These objects can be linked with each other through:
- Links *(see page 827)* or
- actual parameters *(see page 789)* (only FFBs).

Comments for the logic of the program can be added using text objects *(see page 853)*.

## Edit and view functions

Edit and view functions of the LD editor:
- Selecting objects *(see page 727)*
- Deleting objects *(see page 730)*
- Cutting *(see page 730)*, copying *(see page 731)* and pasting *(see page 731)* objects
- Moving *(see page 732)* objects (also between different LD/FBD sections)
- Replacing objects
- Undo *(see page 587)* and Redo *(see page 587)*
- Using bookmarks *(see page 576)*
- Searching and replacing *(see page 1235)* variables and function blocks and DFB instances

## Input assistants

Input assistants for the LD editor:
- Function input assistants *(see page 799)* for functions, function blocks and procedures

## Online functions

Online functions of the LD editor:
- Display the current value *(see page 593)*.
- Setting breakpoints *(see page 1423)*
- Setting watchpoints *(see page 1428)*
- Step by Step *(see page 1425)*

# Creating an LD program

## Introduction

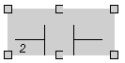
The LD editor window is made up of cells and 1 object can be placed in each one. The cells are separated visually using a grid *(see page 581)* which can be switched off.

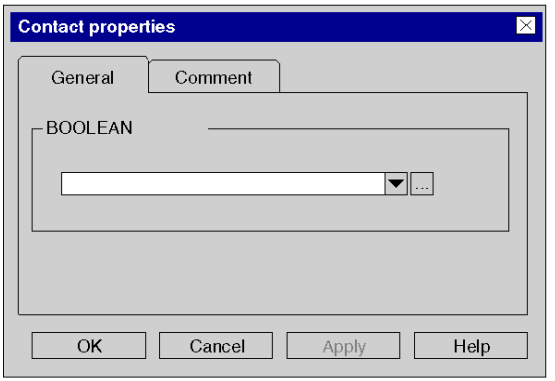
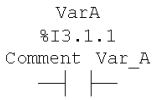
When creating a program with the keyboard, the selected (active) cell has a gray background, see also *Creating an LD program with the keyboard, page 715*.

When creating a program with the mouse, this gray field is not relevant.

## Creating an LD program with the mouse

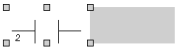
Process the following steps to create an LD program with the mouse:

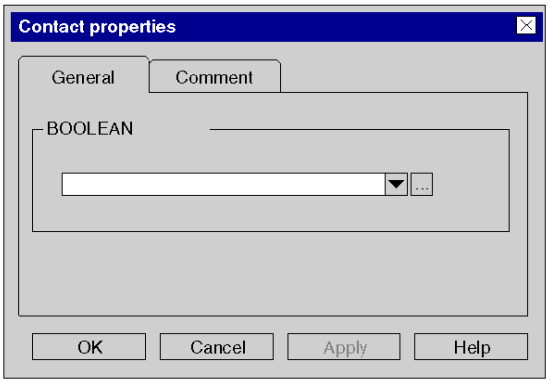
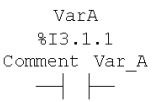
| Step | Action |
|---|---|
| 1 | Create an LD section. *(see page 458)* |
| 2 | Select the desired object (see also *Editing Links, page 827*) using:<br>● the menu commands in the **Edit** → **New** → **<Object>** menu<br>or<br>● the symbols for the objects in the toolbar<br>**Result:** The mouse pointer indicates the selected object (placement mode).<br>**Note:** As soon as the text is entered several checks are made, such as syntax/semantic error checks, correct spelling of variable names etc. A detailed description of syntax is found in chapter Ladder Diagram (LD) *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. The results of the checks are indicated by a colored folder, see also *Syntax and Semantics Check during programming, page 717*. |
| 3 | Click the target cells in the LD section.<br>**Result:** The selected object is inserted and the selection mode is active again.<br>Example:<br>  <br>**Note:** If you want to insert several objects of the same type, press and hold the **Ctrl** key and insert any number of selected objects by clicking with the left mouse button.<br>Return to selection mode by clicking the  symbol or pressing the **Esc** key. |

| Step | Action |
|---|---|
| 4 | To enter the respective actual parameters (variable/address), double-click the object.<br>**Result:** The properties dialog box for the object is opened.<br>Example:<br>  |
| 5 | Enter the desired actual parameter and a comment (if required) and confirm the entries with **OK**.<br>**Result:** In entry mode, the names of the associated variables are displayed above the object. In mixed display mode, the addresses and comments are displayed in addition to the names of the variables (if available), see also *View for variables, page 719*.<br>Example (entry mode):<br>  <br>Example (mixed display mode):<br>  |
| 6 | Repeat these steps until all objects are entered. |

## Creating an LD program with the keyboard

Process the following steps to create an LD program with the keyboard:

| Step | Action |
|---|---|
| 1 | Create an LD section. *(see page 458)* |
| 2 | Place the field with the gray background on the cell where the object should be inserted, see also *Navigating with the Keyboard, page 724*. |
| 3 | Select the desired object (see also *Editing Links, page 827*) using the function keys for the objects.<br>**Result:** The cursor symbol indicates the selected object.<br>**Note:** As soon as the text is entered several checks are made, such as syntax/semantic error checks, correct spelling of variable names etc. A detailed description of syntax is found in chapter Ladder Diagram (LD) *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>The results of the checks are indicated by a color folder, see also *Syntax and Semantics Check during programming, page 717*. |
| 4 | Confirm the selection using the **Enter** key.<br>**Result:** The selected object is inserted into the cell with the gray background and the gray field is automatically shifted to the next cell.<br>Example:<br>  <br>Note: If you want to insert several objects of the same type, press and hold the **Ctrl** key and insert any number of selected objects with the **Enter** key. To return to selection mode, press the **Esc** key. |
| 5 | Press the **Enter** key to open a shortcut menu for the selected object. |

| Step | Action |
|---|---|
| 6 | To enter the respective actual parameters (variable/address), use the **Properties** menu command from the shortcut menu.<br>**Result:** The properties dialog box for the object is opened.<br>Example:<br>  |
| 7 | Enter the desired actual parameter and a comment (if required) and confirm the entries with **OK**.<br>**Result:** In entry mode, the names of the associated variables are displayed above the object. In mixed display mode, the addresses and comments are displayed in addition to the names of the variables (if available), see also *View for variables, page 719*.<br>Example (entry mode):<br>  <br>Example (mixed display mode):<br>  |
| 8 | Repeat these steps until all objects are entered. |

# Syntax and Semantics Check during programming

## Introduction

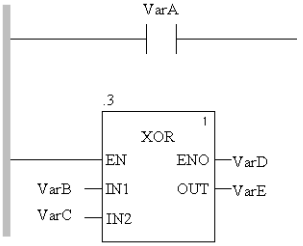
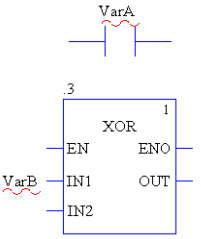
Syntax and semantics check is performed directly when creating the program.

The result of this check is displayed in three formats:

- directly in the program section with colored text
- as tooltip if the cursor is placed on incorrect text.
- In the output window, if **Generate** → **Analyze** is selected.

## Representation

Meaning of colors and labels:

| Color | Description | Example |
|---|---|---|
| Black | Syntax and semantics are correct |  |
| Blue | Possible causes:<br>● Associated variable is not declared<br>● Data type of the variable does not match the data type of the pin<br>● Input or output pin not connected |  |
| Red wavy line | Faulty text, e.g. variables which are not declared, variables with an incorrect data type, ... |  |

## Tooltips

If the cursor is placed over incorrect text, the tooltip is displayed with a brief description of the cause of the error. It will also be displayed in the output window after the analysis.

## Error message in the output window

Error message: `Accessed object is not a function block`.

| Cause of error | Error Correction |
|---|---|
| Name of the function block was already used for a variable. | Rename the already used variable. |

# View for variables

## Display modes for variables

The following view modes are available for variables:

- Input mode *(see page 720)*
- Mixed display mode *(see page 721)*

You can switch between the two modes using:

- The **View → Mixed Display Mode** menu command
- The **Ctrl**+**E** key combination.

A check-symbol displayed on the left of the **View → Mixed Display Mode** item indicates when the mixed display mode *(see page 721)* is enabled.

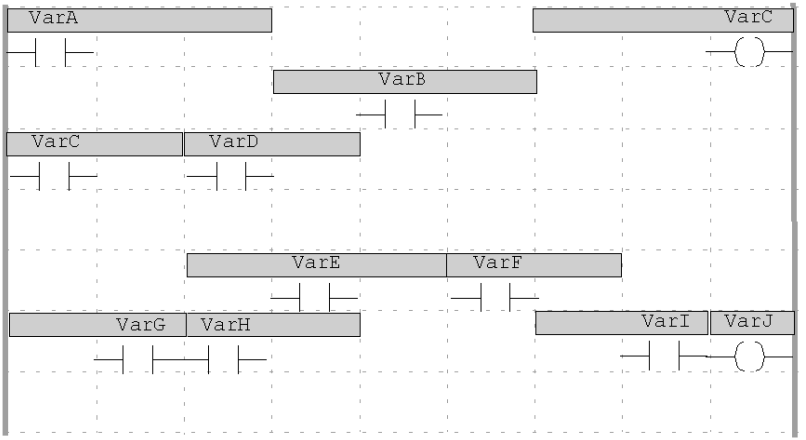
## Variables display

Depending on the size set for the cells, it may be the case that variable names are not shown in full.

A maximum of 3 columns are available to display the variables. The space available depends on various factors, on both Input mode *(see page 720)* & Mixed display mode *(see page 721)*

- On the position
  - If an element (e.g. a contact) was placed directly on the left power rail, columns 1-3 are available for the variable names.
  - If an element (e.g. a coil) was placed directly on the right hand power rail, columns 9-11 are available for the variable names.
  - If an element (e.g. a contact) was placed arbitrarily in the section, the column to the right and the column to the left of the element are available for the variable name.
- On the availability of the cells:
  - Only cells that are not occupied by other objects are available to display the variable name.
  - If an unused cell is between two objects, it is automatically reserved for the variable name of the left hand object and is not available to the right hand object (even if it is not used by the left hand object).

The gray fields indicate which area is available for the variable names of the individual objects.



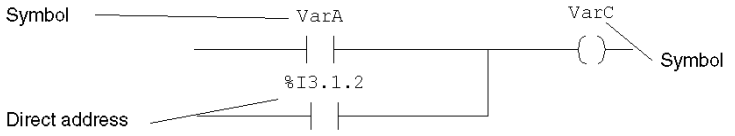

**Tip:** There are two options for displaying the full variable names:
- Select the variable and the full variable name is shown in the status line.
- Place the cursor on the variable and a Tooltip will appear with type, name, address and comment of the variable.

## Input mode

In input mode, only the symbolic name is displayed for variables and only the address for direct addresses:

- if you enter a symbol you will see in this mode a symbol even if you associate an address after in the data editor.
- if you enter a directe address you will see in this mode a direct address even if you associate a symbol after in the data editor.

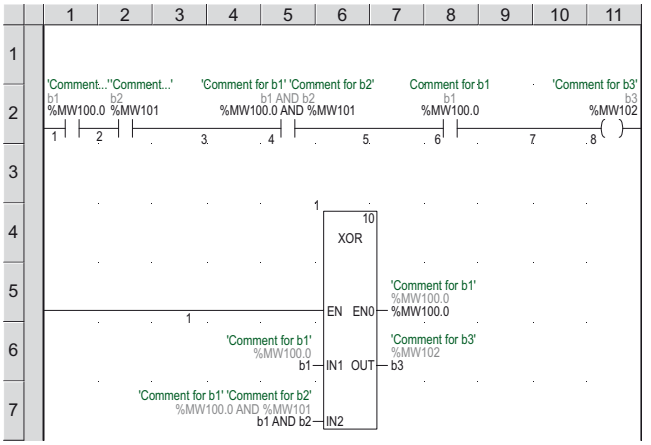
Example of a section in input mode:



## Mixed display mode

In the mixed display mode, the comment, symbolic name and, if a unique association is possible, the address will be displayed for the variables.

Example of a section in Mixed Display Mode:




This functionality is available for variables for both LD objects (contact, coil, comparative module, operation module) and FFBs.

**NOTE:** When using structure elements, the entire comment (comment for root variables and comment(s) for the structure element(s)) or only the comment for the root variable may be displayed. To display the entire comment, select **Tools → Project Settings → Editors tab** and enable **Display complete comments for structure elements**.

**NOTE:** To print one or several sections the option `mixed display` must be check in **Documentation configuration** of project.

For example, the following displays are possible:

| Display | Description |
|---|---|
| `'VariableA'`<br>`VarA`<br>`%I3.1.2` | The located variable `VarA` with the comment `VariableA` has the address `%I3.1.2`.<br>For this type of display the following requirements apply:<br>● The display of comments is enabled.<br>● The variable is a located variable.<br>● A comment was defined for the variable.<br>● An address is only associated with one symbolic name<br>or<br>the symbolic name was assigned to the actual parameter. |

| Display | Description |
|---|---|
| 'Mounting.ManualStart'<br>Motor.Start<br>%I3.1.2 | The structure element `Start` of the variable `Motor` is assigned to the address `%I3.1.2` assigned. The displayed comment consists of the comments for the root variable (`Motor`) and the comment(s) for the structure elements (`Start`).<br>For this type of display the following requirements apply:<br>● In **Tools → Project Settings → Editors tab** the option **Display complete comments for structure elements** is enabled.<br>● The variable is a located variable.<br>● A comment was defined for the variable.<br>● An address is only associated with one symbolic name<br>or<br>the symbolic name was assigned to the actual parameter. |
| 'Mounting'<br>Motor.Start<br>%I3.1.2 | The element `Start` of the variable `Motor` is assigned to the address `%I3.1.2` assigned. The displayed comment is the comment for the root variable (`Motor`).<br>For this type of display the following requirements apply:<br>● In **Tools → Project Settings → Editors tab** the option **Display complete comments for structure elements** is disabled.<br>● The variable is a located variable.<br>● A comment was defined for the variable.<br>● An address is only associated with one symbolic name<br>or<br>the symbolic name was assigned to the actual parameter. |
| '<n/a>'<br>VarA<br>%I3.1.2 | The located variable `VarA` has the address `%I3.1.2`and no comment was defined for the variable.<br>For this type of display the following requirements apply:<br>● The variable is a located variable.<br>● An address is only associated with one symbolic name<br>or<br>the symbolic name was assigned to the actual parameter. |

| Display | Description |
|---|---|
| 'VariableA'<br>VarA<br><n/a> | The variable VarA is an unlocated variable. |
| '<n/a>'<br><...><br>%I3.1.2 | An address with multiple assigned symbolic names was assigned to the object as an actual parameter. Since in this case, a unique assignment of symbolic name to address is not possible, no symbolic name is displayed. |
| 'VariableB'<br><n/a><br>%I3.1.2 | The address was a assigned a comment, but no symbolic name. |
| '<n/a>'<br>VarA:VarB<VarC<br><###><br>or<br>'<n/a>'<br><###><br>%I3.1.2:%I3.1.3<%I3.1.4 | Syntax error. The expression entered is not correct. For this reason no addresses or symbols can be displayed. |

**NOTE:** The cell height and column width available to display additional information can be specified for the current project under **Editors** for the Dialog **Tools → Project Settings** and for future projects under **Data and Languages** of the Dialog **Tools → Options**.

# Navigating with the Keyboard

## Using the keyboard

The following keys and key combinations are provided for navigation:

| Key combinations | Motion |
|---|---|
| **Cursor left** | Moves the gray field in a cell to the left by one cell and selects the contents of that cell.<br>If the first line is reached, the previous line is automatically selected. |
| **Cursor right** | Moves the gray field in a cell to the right by one cell and selects the contents of that cell.<br>If the last line is reached, the next line is automatically selected. |
| **Cursor up** | Moves the gray field in a cell to the up by one cell and selects the contents of that cell |
| **Cursor down** | Moves the gray field in a cell to the down by one cell and selects the contents of that cell |
| **Ctrl+Left** | Moves the cursor 1 pixel left and moves the gray field to the cursor position |
| **Ctrl+Right arrow** | Moves the cursor 1 pixel right and moves the gray field to the cursor position |
| **Ctrl+Up arrow** | Moves the cursor 1 pixel up and moves the gray field to the cursor position |
| **Ctrl+Down arrow** | Moves the cursor 1 pixel down and moves the gray field to the cursor position |
| **Ctrl+Shift+Left arrow** | Moves the cursor one cell left |
| **Ctrl+Shift+Right arrow** | Moves the cursor one cell right |
| **Ctrl+Shift+Up** | Moves the cursor one cell up |
| **Ctrl+Shift+Down arrow** | Moves the cursor one cell down |
| **Shift+Left arrow** | Moves the selected object and the gray field to the left by one cell<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |
| **Shift+Right arrow** | Moves the selected object and the gray field to the right by one cell.<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |
| **Shift+Up arrow** | Moves the selected object and the gray field up by one cell.<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |

| Key combinations | Motion |
|---|---|
| **Shift**+**Down arrow** | Moves the selected object and the gray field down one cell.<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |
| **Home** | Positions the gray field in the first column and shows this new location. |
| **End** | Positions the gray field in the last column and shows this new location. |
| **Ctrl**+**Home** | Positions the gray field in the upper left hand cell in this section and shows this new location. |
| **Ctrl**+**End** | Positions the gray field in the upper right hand cell in this section and shows this new location. |
| **Page Up** | Scrolls the position of the gray field one page up and shows this new position. |
| **Page Down** | Scrolls the position of the gray field one page down and shows this new position. |
| **Ctrl**+**Page Up** | Scrolls the position of the gray field one page left and shows this new position. |
| **Ctrl**+**Page Down** | Scrolls the position of the gray field one page right and shows this new position. |
| **Ctrl**+**Alt**+**Page Up** | Displays the previous section (including via the **View** → **Previous Section**menu). |
| **Ctrl**+**Alt**+**Page Down** | Displays the next section (including via the **View** → **Next Section**menu). |
| **Spacebar** | Selects or deselects the object in the cells highlighted in gray.<br>If several objects are in the gray highlighted field, the next object in the cell is selected every time you press the **space bar**.<br>The position of the gray field is not affected by this function.<br>See also *Selecting objects, page 727*. |
| **Ctrl** + **space bar** | If several objects are in the gray highlighted cells, every time you press the shortcut key combination **Ctrl**+**space bar** the next object is inserted in the currently selected cells (multiple selection).<br>The position of the gray field is not affected by this function. |
| **Shift** + **space bar** | The position of the gray field is moved one cell to the right and all objects in the currently selected cells highlighted in gray (multiple selections) are inserted. |

| Key combinations | Motion |
|---|---|
| **Enter** | In Select mode: Opens the properties dialog box for the selected contact / coils. When you are finished press **Enter** to cancel press **Esc**.<br>In Insert mode: Inserts the selected object into the currently selected cells highlighted in gray and moves the position of the gray field one cell to the right. |
| **Alt+Enter** | Opens the properties dialog box for the selected object/pin. |
| **Tab** | Select the next pin in an FFB if the FFB or an FFB pin is selected. |
| **Esc** | Activates Select mode. |

# Selecting objects

## Select and placement mode

Objects are selected in select mode.

Select mode can be activated using:

- the **Edit** → **Select Mode** menu command
- the  symbol,
  or
- the **Esc** key

The  cursor symbol indicates that select mode is active

## Selecting an object

| Using the mouse | Using the keyboard |
|---|---|
| Left-click the object you want to select. | Move the gray field to the cell with the object to be selected, see also *Navigating with the Keyboard, page 724*.<br>**Note:** If several objects are in the gray highlighted field, the next object in the cell is selected every time you press the **spacebar**. |

## Selecting several objects

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Click the left mouse button and keep it pressed.<br>**2.** Drag the mouse across the objects you want to select.<br>or<br>**1.** Left-click the first object you want to select.<br>**2.** Press the **Ctrl** key and keep it pressed.<br>**3.** Left-click the next object you want to select.<br>**4.** Repeat these steps until all desired objects are selected. | Selecting several objects in **different** cells:<br>**1.** Move the gray field to the cell with the first object to be selected, see also *Navigating with the Keyboard, page 724*.<br>**2.** Press the **Shift**+**spacebar** key combination.<br>**Result:** The object is added to the current selection and the gray field is shifted one cell to the right.<br>**3.** Repeat these steps until all desired objects are selected.<br>Selecting several objects in **one** cell:<br>**1.** Move the gray field to the cell with the objects to be selected, see also *Navigating with the Keyboard, page 724*.<br>**2.** Press the **spacebar** until the first desired object is selected.<br>**3.** Press the **Ctrl**+**spacebar** key combination to select the next object in the cell.<br>**4.** Repeat these steps until all desired objects are selected. |

## Selecting by rows

| Using the mouse | Using the keyboard |
|---|---|
| Selecting the contents of a row:<br>**1.** Left-click the number of the row you want to select in the vertical ruler.<br>Selecting the contents of several lines:<br>**1.** Left-click the number of the first line you want to select in the vertical ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last line you want to select in the vertical ruler.<br>**Note:** Pressing the **Shift** key makes it possible to combine this procedure with the procedure for selecting columns. | - |

## Selecting by columns

| Using the mouse | Using the keyboard |
|---|---|
| Selecting the contents of a column:<br>**1.** Left-click the number of the column you want to select in the vertical ruler.<br>Selecting the contents of several columns:<br>**1.** Left-click the number of the first column you want to select in the vertical ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last column you want to select in the vertical ruler.<br>**Note:** Pressing the **Shift** key makes it possible to combine this procedure with the procedure for selecting lines. | - |

## Selecting all (entire contents of the section)

| Using the mouse | Using the keyboard |
|---|---|
| Use the **Edit** → **Select All** menu command. | Press the **Ctrl**+**A** key combination. |

## Selecting all objects

| Using the mouse | Using the keyboard |
|---|---|
| Left-click an empty space in the section. | **1.** Move the gray field to an empty space, see also *Navigating with the Keyboard, page 724*.<br>**2.** Press the **spacebar**. |

## Deselecting objects

| Using the mouse | Using the keyboard |
|---|---|
| Left-click an empty space in the section. | Move the gray field or press **the space bar**. |

# Deleting, cutting, copying, pasting and moving objects

## Deleting objects

Deleting objects:

| using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 727)* the object to be deleted.<br>2. Use the **Edit** → **Delete** menu command. | Deleting the selected object:<br>1. Select *(see page 727)* the object to be deleted.<br>2. Press the **Entf** key.<br><br>Deleting the object left of the gray field:<br>1. Press the **Backspace** key. |
| **Result:** The selected object is deleted. The actual parameters (and the links for FFBs) are deleted together with the object, even if they were not selected explicitly. Boolean links are only deleted if they are selected explicitly. | |

## Cutting objects

Cutting objects:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 727)* the object to be cut.<br>2. Use the **Edit** → **Cut** menu command.<br>or<br>Use the menu command **Cut** in the shortcut menu (right mouse button).<br>or<br>Click the  symbol. | 1. Select *(see page 727)* the object to be cut.<br>2. Press the **Ctrl**+**X** key combination. |
| **Result:** The selected object is cut from the section and copied to the clipboard. This is also the case for the actual parameters. FFB links are also cut. But they are only copied to the clipboard if they (and their partner objects) are selected explicitly. | |

The cut object can be inserted *(see page 731)* in any other position (also in another LD section).

## Copying objects to the clipboard

Copying objects to the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 727)* the object to be copied.<br>2. Use the **Edit** → **Copy** menu command.<br>or<br>Use the **Copy** menu command from the shortcut menu (right-click).<br>or<br>Click the  symbol. | 1. Select *(see page 727)* the object to be copied.<br>2. Press the **Ctrl**+**C** key combination. |
| **Result:** The selected object is copied to the clipboard. This is also the case for the actual parameters. FFB links are only copied to the clipboard if they (and their partner objects) are selected explicitly. | |

The cut object can be inserted *(see page 731)* in any other position (also in another LD section).

## Pasting objects from the clipboard

Pasting objects from the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Use the menu command **Edit** → **Insert**.<br>or<br>Use the menu command **Paste** in the shortcut menu (right mouse button).<br>or<br>Click the  symbol.<br>2. Left-click the target position. | 1. Press the **Ctrl**+**V** key combination.<br>2. Move the cursor to the target position using the arrow keys.<br>3. Press **Enter**. |
| **Note for pasting contacts and coils:** If the target position already contains a contact or coil, the existing object is overwritten by the new object.<br>**Note:** This presents the behavior of the **Copy**, **Cut**, **Paste** on a function block instance. It only applies to graphical languages as **FBD** and **LD**.<br>● Using the **Paste** function after a **Copy** of an object:<br>As a result of a **Copy**, a new function block instance (**FBI**) is used. The **Paste** function creates a new FBI whenever it is repeated. Accordingly, the FBI is incremented.<br>● Using the **Paste** function after a **Cut** of an object:<br>As a result of a **Cut**, the same instance of the function is used. The **Paste** function uses the same FBI whenever it is repeated. Accordingly, the FBI is identical. | |

## Moving objects

Moving objects:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 727)* the object to be moved.<br>**2.** Position the mouse pointer on the selected object. (on one of the selected objects if several are selected).<br>**Result:** The mouse pointer changes its symbol to  .<br>**3.** Click the left mouse button and keep it pressed.<br>**Result:** The mouse pointer changes its symbol to  .<br>**4.** Drag the object to the new position and release the mouse button. | **1.** Position *(see page 727)* the gray field on the object to be moved.<br>**2.** Press the **spacebar**.<br>**3.** Press the **Shift** key and keep it pressed.<br>**4.** Move the object to the target position using the arrow keys. |
| **Result:** The selected object is moved from its original position to the target position. The object's actual parameters are moved together with the object. FFBs still have their FFB-links and they are adjusted to the new position of the FFB. Boolean links are only kept for horizontal movement operations.<br>**Note:** A moving of FFBs to cells that are already occupied (e.g. by contact, coil, other FFB, boolean link), is not possible.<br>**Note:** The move operation is also possible across sections between different opened LD sections.<br>**Note:** When objects are moved beyond the visible area of the editor window, automatic scrolling of the window will not begin until you reach the edge of the window with the mouse pointer, not as soon as the objects are moved beyond the edge. | |

## Copying objects via drag & drop

Copying objects via drag & drop:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 727)* the object to be copied.<br>**2.** Position the mouse pointer on the selected object.<br>**Result:** The mouse pointer changes its symbol to  .<br>**3.** Click the left mouse button and keep it pressed.<br>**4.** Press the **Ctrl** key and keep it pressed.<br>**Result:** The mouse pointer changes its symbol to  .<br>**5.** Drag the object to the new position.<br>**Note:** This is also possible across sections between different opened LD sections.<br>**6.** Release the mouse button.<br>**Result:** A copy of the selected object is pasted at the target position. The object's actual parameters (variable/address) are copied together with the object. A new instance is automatically created for FFBs. FFB links are only copied if they (and their partner objects) are selected explicitly. | - |

# Displaying the properties

## Displaying the object properties

You have the following options to display the object properties dialog box:

- Double-click the object.
- Select *(see page 727)* the object and use the **Edit → Properties... dialog box.**.
- Select *(see page 727)* the object and use the **Properties...command**from the shortcut menu.
- Select *(see page 727)* the object and then press the **Alt**+**Enter** key combination.

When the properties dialog box is called without an object being selected, the section properties dialog box *(see page 461)* is opened.

If several objects are selected, the commands do not work.

## View data properties

You have the following options to display the section properties dialog box *(see page 1197)*:

- using the data editor *(see page 325)*
  **a.** Selecting one or several lines in a list editor
  **b.** Using the shortcut menu select the **Properties**command.
- via the LD section
  **a.** Selecting one or several element(s) in the LD section.
  **b.** From the context menu select the command **Data properties** or press **Ctrl** + **Enter**.

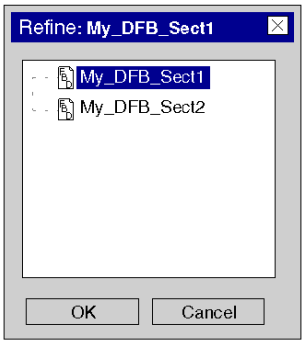
# Refining DFBs and subroutines

## Introduction

This function makes it possible to "look into" a DFB or subroutine. This makes it possible to see the internal state of the DFB/subroutine when animation is used, or to make changes to the logic very quickly.

## Refining DFBs

Process the following steps to refine DFBs:

| Step | Action |
|---|---|
| 1 | Select the DFB. |
| 2 | You have the following options to refine DFBs:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** If the DFB contains only one section, this will be opened automatically. If the DFB contains several sections, a selection dialog is opened.<br>Example:<br>  |
| 3 | Select the desired DFB section.<br>**Result:** The DFB section is opened.<br>**Note:** With nested DFBs, several **refine** processes are possible. |

## Refining subroutines

Process the following steps to refine subroutines:

| Step | Action |
|---|---|
| 1 | Select the subroutine block *(see page 674)*. |
| 2 | You have the following options to to refine subroutines:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** The subroutine section is opened.<br>**Note:** With nested subroutines several **refine** processes are possible. |

# Inserting and Deleting Rows

## Rules for Inserting Rows

The following rules apply when inserting rows:

- The same number of rows are inserted as are selected in the vertical ruler.
- The new rows are inserted above the first selected row.
- The first selected row may not contain any objects.

**NOTE:** The number of columns must be specified in the **Tools** → **Project Settings** dialog box, on the **Editors** tab, in the **Number of columns** text box before the LD section is created. The number of columns for LD sections cannot be changed later.

## Inserting Rows

| Step | Action |
|---|---|
| 1 | Select *(see page 728)* the desired number of rows in the vertical ruler (click the row numbers). |
| 2 | Use the **Insert Row** menu command from the shortcut menu of the vertical ruler or use the **Edit** → **Insert Row** menu command.<br>**Result:** The selected number of rows is inserted above the first selected row. |

## Rules for Deleting Rows

The following rules apply when deleting rows:

- The rows to be deleted may not contain any objects.

**NOTE:** The number of columns must be specified in the **Tools** → **Project Settings** dialog box, on the **Editors** tab, in the **Number of columns** text box before the LD section is created. The number of columns for LD sections cannot be changed later.

## Deleting Rows

| Step | Action |
|---|---|
| 1 | Select *(see page 729)* the desired number of rows to be deleted from the vertical ruler (click the row numbers). |
| 2 | Use the **Delete Row** menu command from the shortcut menu of the vertical ruler or use the **Edit** → **Delete Row** menu command.<br>**Result:** The selected rows are deleted. |

# Section 22.2
# Editing Contacts

## Overview

This section describes how to edit contacts in the LD programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Select contacts | 739 |
| Placing contacts | 741 |
| Property dialog box for contacts | 745 |

# Select contacts

## Introduction

A contact is an LD element that transfers a status on the horizontal link to its right side. This status comes from the Boolean AND link of the status of the horizontal link on the left side with the status of the relevant Boolean current parameter.

## Selecting contacts

Selecting contacts:

| Contact type | Description | Call via the menu command | Call via the icon | Call via a key | Cursor symbol |
|---|---|---|---|---|---|
| N.O. contact | For normally open contacts, the status of the left link is copied to the right link, if the status of the relevant Boolean actual parameter is 1. Otherwise, the status of the right link is 0. | **Edit** → **New** → **N.O. Contact** or **Contacts** → **N.O. Contact** in the shortcut menu |  | **F3** |  |
| N.C. contact | For normally closed contacts, the status of the left link is copied to the right link, if the status of the relevant Boolean actual parameter is 0. Otherwise, the status of the right link is 0. | **Edit** → **New** → **N.C. Contact** or **Contacts** → **N.C. Contact** in the shortcut menu |  | **Shift+F3** |  |
| Positive transition contact | With contacts for detection of positive transitions, the right link for a program cycle is 1 if a transfer of the relevant actual parameter goes from 0 to 1 and the status of the left link is 1 at the same time. Otherwise, the status of the right link is 0. Also see the Edge recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the user guide. | **Edit** → **New** → **Positive transition contact** or **Contacts** → **Positive transition contact** in the shortcut menu |  | **Ctrl+F3** |  |

| Contact type | Description | Call via the menu command | Call via the icon | Call via a key | Cursor symbol |
|---|---|---|---|---|---|
| Negative transition contact | With contacts for detection of negative transitions, the right link for a program cycle is 1 if a transfer of the relevant actual parameter goes from 1 to 0 and the status of the left link is 1 at the same time. Otherwise, the status of the right link is 0.<br>Also see the Edge recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the user guide. | **Edit → New → Negative transition contact**<br>or<br>**Contacts → Negative transition contact** in the shortcut menu |  | **Ctrl** + **Shift** + **F3** |  |

# Placing contacts

## Introduction

Contacts can be placed in any free cell except cells directly on the right power rail.

If a contact is placed in a cell that was previously occupied by a contact, coil, a Boolean link or a label, the cell contents are replaced by the new contact.

If a contact is placed in a cell that is already occupied by another object (e.g. FFBs), an error message is returned.

A placed contact automatically creates a connection with its neighboring objects on the left and right if they are of the `Bool` data type and there are no free cells between them.

The following are permitted as actual parameters for contacts:

- Boolean variables
- Boolean literals (`0`, `1`, `FALSE`, `TRUE`)
- Boolean addresses (topological addresses or symbolic addresses)
- ST expressions that return a Boolean result (e.g. `VarA > VarB`)
  ST expressions as formal parameters on contacts are an extension of IEC 61131-3 and must be explicitly enabled via the **Tools → Project Settings** dialog box, in the **Language extensions** tab by selecting the **Usage of ST expressions** checkbox.

## Note

In the **Tools → Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the checkbox is **cleared**.

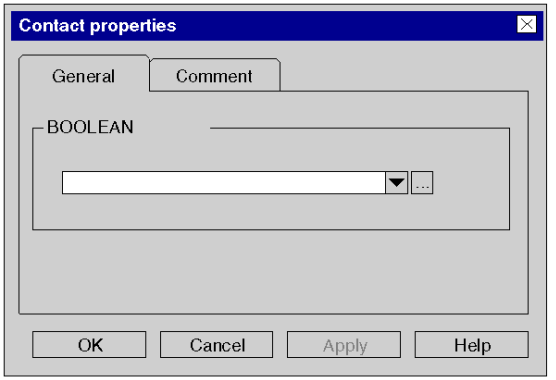
## Placing contacts:

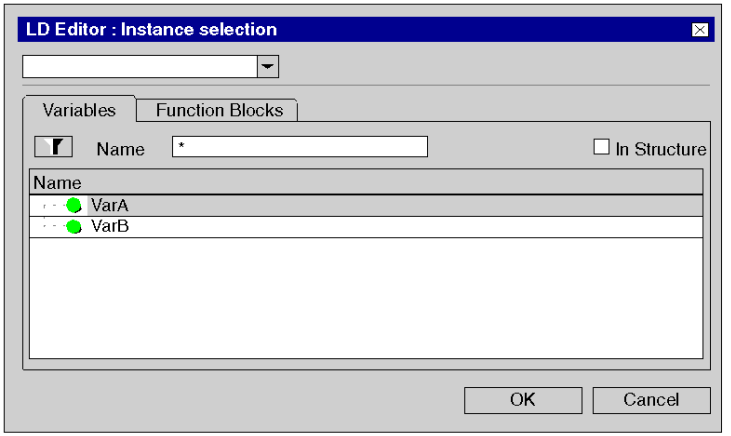
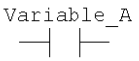
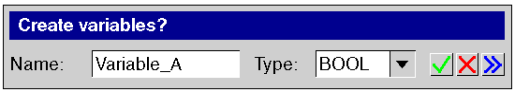
Placing contacts:

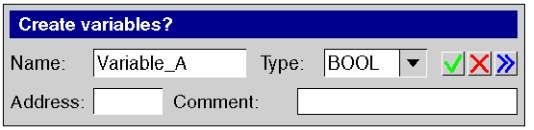
| Step | Action |
|---|---|
| 1 | Select the desired contact, see also *Selecting contacts, page 739*. |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position, and press **Enter**.<br>**Result:** The selected contact is inserted. |
| 3 | To place further contacts of the same type:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Assigning actual parameters

Assigning actual parameters:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 734)* for the contact.<br>**Result**: The contacts properties dialog box is opened, see also *Calling the properties dialog box, page 745*.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | You have the following options to enter the actual parameters:<br>● You can now enter the actual parameter and confirm using **Enter**.<br>● You can select the actual parameter from the list of recently used names using the ▾ symbol and confirm using **Enter**.<br>or<br>● Using the button **...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example of a variable selection dialog box:<br>  |
| 3 | Confirm the selected variable with **OK**.<br>**Result:**<br>● If the selected variable is declared, it is enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>  <br>● If the variable is not declared, a variable declaration dialog box opens. In this case, continue with the following steps of this procedure.<br>Variable declaration dialog box<br>  <br>**Note:** If you wish to declare the variable later, close the dialog box using the ☒ symbol. In this case, the variable name is enabled in the section, but not declared. |

| Step | Action |
|---|---|
| 4 | If you do not want to assign the variable an address or comment, confirm using **Enter** or the symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>Example:<br>Variable_A |
| 5 | If you do want to assign the variable an address and/or comment, use the symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 6 | Confirm with **Enter** or the symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check *(see page 717)* is performed immediately after the statement is entered.<br>Example:<br>Variable_A |

## Usage of ST expressions

To enter ST expressions for contacts the same rules apply as for entering ST expressions for FFB pins. *(see page 798)*

# Property dialog box for contacts

## Calling the properties dialog box

see *Displaying the properties, page 734*

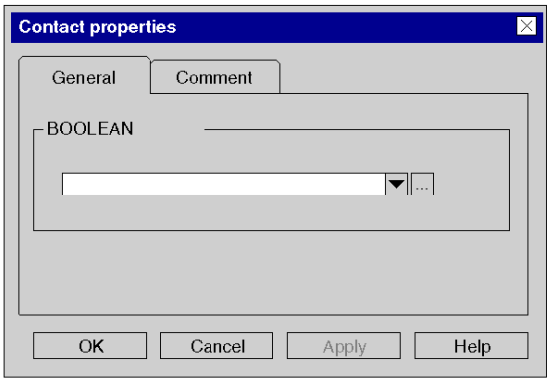
## Structure of the properties dialog box

The contact properties dialog box consists of two tabs:

- **General**
  In this tab, you can enter the actual parameter *(see page 742)* of the contact.
- **Comment**
  In this tab, a comment about the contact can be entered.

## General tab

Representation of the **General** tab:

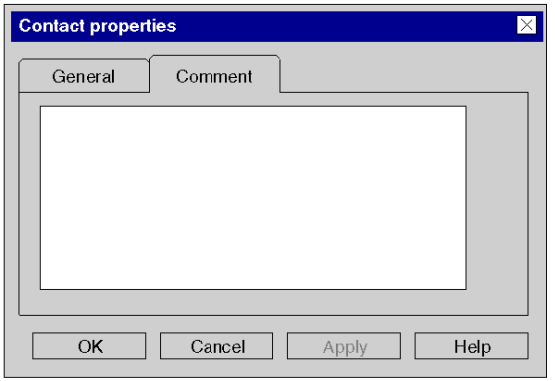


Elements of the **General** tab:

| Element | Description |
|---|---|
| **BOOLEAN Expression** | Enter the name of the actual parameter in this text box.<br>You have the following options:<br>● You can directly enter the name of the variable/address or paste it from the clipboard.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Use the command button **...**to open a variable selection dialog box *(see page 1182)*.<br><br>The actual parameters which are permitted are:<br>● Boolean variables<br>● Boolean constants<br>● Boolean addresses (topological addresses or symbolic addresses)<br>● ST expression delivering a Boolean result (e.g. `BoolVar1 OR BoolVar1`)<br>● Literal (`1` or `0` or. `TRUE` or `FALSE`) |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:



Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the contact.<br>This comment is displayed as a tooltip when the cursor is placed over the contact. It is independent from the variable comment that can be assigned to the variable in the data editor. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 22.3
# Editing Coils

## Overview

This section describes how to edit coils in the LD programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selection of coils | 749 |
| Placing coils | 752 |
| Property dialog box for coils | 758 |

# Selection of coils

## Introduction

A coil is an LD element which transfers the status of the horizontal link on the left side, unchanged, to the horizontal link on the right side. The status is saved in the respective Boolean actual parameter. Coils normally follow contacts or FFBs, (functions and function blocks), but they can also be followed by contacts.

## Selecting coils

Selecting coils:

| Coil type | Description | Call via menu command | Call via symbol | Call via keyboard | Cursor symbol |
|---|---|---|---|---|---|
| Coil | With coils, the status of the left link is copied to the relevant Boolean actual parameter and the right link. | **Edit** → **New** → **Coil**<br>or<br>**Coils** → **Coil** from the shortcut menu |  | **F5** |  |
| Negated coils | With negated coils, the status of the left link is copied onto the right link. The inverted status of the left link is copied to the relevant Boolean actual parameter. If the left link is 0, the right link is also 0 and the respective Boolean actual parameter is 1. | **Edit** → **New** → **Negated Coil**<br>or<br>**Coils** → **Negated Coil** from the shortcut menu |  | **Shift+F5** |  |
| Positive transition-sensing coil | With coils for detection of positive transitions, the status of the left link is copied to the right link. The relevant Boolean actual parameter is 1 for a program cycle, if a transition from 0 to 1 is made on the left link.<br>Also see the Edge recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the user guide. | **Edit** → **New** → **Positive transition-sensing coil**<br>or<br>**Coils** → **Positive transition-sensing coil** from the shortcut menu |  | **-** |  |

| Coil type | Description | Call via menu command | Call via symbol | Call via keyboard | Cursor symbol |
|---|---|---|---|---|---|
| Negative transition-sensing coil | With coils for detection of negative transitions, the status of the left link is copied to the right link. The relevant Boolean actual parameter is 1 for a program cycle, if a transition from 1 to 0 is made on the left link.<br>Also see the Edge recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the user guide. | **Edit → New → Negative transition-sensing coil**<br>or<br>**Coils → Negative transition-sensing coil** from the shortcut menu |  | **-** |  |
| Set coil | With set coils, the status of the left link is copied onto the right link. The relevant Boolean actual parameter is set to 1 if the left link has the status 1, otherwise it remains unchanged. The relevant Boolean actual parameter can be reset through the reset coil. | **Edit → New → Set Coil**<br>or<br>**Coils → Set Coil** from the shortcut menu |  | **Alt+F5** |  |
| Reset coil | With reset coils, the status of the left link is copied onto the right link. The relevant Boolean actual parameter is set to 0 if the left link has the status 1, otherwise it remains unchanged. The relevant Boolean actual parameter can be set through the set coil. | **Edit → New → Reset Coil**<br>or<br>**Coils → Reset Coil** from the shortcut menu |  | **Shift + Alt + F5** |  |
| Halt coil | With halt coils, if the status of the left link is 1, the program execution is stopped immediately. (With halt coils the status of the left link is not copied to the right link.) | **Edit → New → Halt coil**<br>or<br>**Coils → Halt coil** from the shortcut menu |  | **-** |  |

| Coil type | Description | Call via menu command | Call via symbol | Call via keyboard | Cursor symbol |
|---|---|---|---|---|---|
| Call coil | With call coils, the status of the left link is copied to the right link. If the status of the left link is 1, the corresponding subroutine is called.<br>Subroutine calls are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by selecting the **Allow subroutines** check box. | **Edit** → **New** → **Call Coil**<br>or<br>**Coils** → **Call Coil** from the shortcut menu |  | **F4** |  |

# Placing coils

## Introduction

Coils can be placed in any free cell except cells directly on the left power rail.

If all coils are to be placed on the right power rail, select the **Tools → Project Settings** menu, and in the **Editors** tab, check the **Right-justify coils** checkbox.

If a coil is placed in a cell that is already occupied by a coil, contact, a Boolean link or a label, the cell contents are replaced by the new coil.

If a coil is placed in a cell that is already occupied by another object (e.g. FFBs), an error message is returned.

A placed coil automatically creates a connection with its neighboring objects on the left and right if they are of the `BOOL` data type, even if free cells are between them.

The following are permitted as actual parameters for coils:
- Boolean variables
- Boolean addresses (topological addresses or symbolic addresses)

## Note

In the **Tools → Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the checkbox is **cleared**.

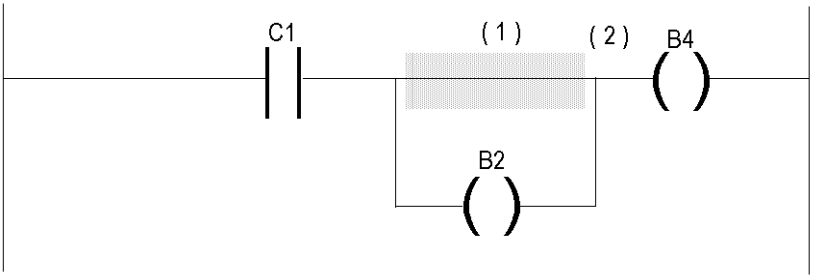
## Placing Coils

Placing Coils

| Step | Action |
|---|---|
| 1 | Select the desired coil, see also *Selecting coils, page 749*. |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position, and press **Enter**.<br>**Result:** The selected coil is inserted.<br>In the **Tools → Project Settings** dialog box, on the **Editors** tab, if the **Right-justify coils** checkbox is selected and the last cell in the current row (cell directly on the right power rail) is not occupied, the coil is placed automatically on the right power rail. If this cell is already occupied, the coil is inserted into the current cell (cursor position gray field). |

| Step | Action |
|---|---|
| 3 | To place further coils of the same type:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

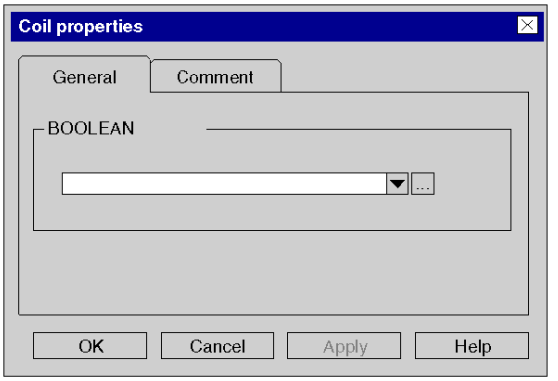
**NOTE:** When editing a program in Ladder, a short circuit on a coil may lead to a state always true. For a good logic behavior, the short-circuit must be deleted..

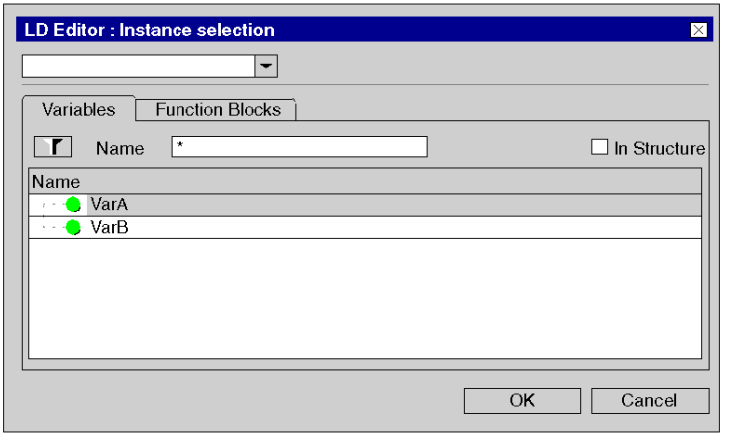
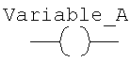


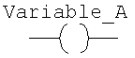
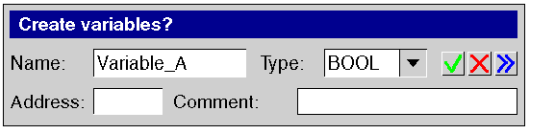
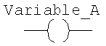

If the short-circuit (1) is located on top of the coil B2, this coil will remain in a true state (B2 = 1) regardless of the logic activating the coil C1. As a consequence, the active state (2) is propagated to the following logic (b4 = 1).

## Assigning actual parameters

Assigning actual parameters:

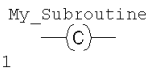
| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 734)* for the coil.<br>**Result**: The coils properties dialog box is opened, see also *Property dialog box for coils, page 758*.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | You have the following options to enter the actual parameters:<br>● You can now enter the variable/address name and confirm using **Enter**.<br>● You can select the variable name from the list of recently used names using the ▾ symbol and confirm using **Enter**.<br>or<br>● Using the button **...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example of a variable selection dialog box:<br>  |
| 3 | Confirm the selected variable with **OK**.<br>**Result:**<br>● If the selected variable is declared, it is enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>  <br>● If the variable is not declared, a variable declaration dialog box opens. In this case, continue with the following steps of this procedure.<br>Variable declaration dialog box<br>  <br>**Note:** If you wish to declare the variable later, close the dialog box using the ☒ symbol. In this case, the variable name is enabled in the section, but not declared. |

| Step | Action |
|---|---|
| 4 | If you do not want to assign the variable an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>Example:<br>  |
| 5 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 6 | Confirm with **Enter** or the  symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  |

## Calling a subroutine

Calling a subroutine:

| Step | Action |
|---|---|
| 1 | Place a call coil in the section, see also *Selecting coils, page 749*.<br>**Note:** Subroutine calls are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by selecting the **Allow subroutines** checkbox. |
| 2 | Open the properties dialog box *(see page 734)* for the coil.<br>**Result**: The coils properties dialog box is opened, see also *Property dialog box for coils, page 758*. |
| 3 | Click the ▾ symbol and select the subroutine to call from the list and confirm using **Enter**.<br>**Note:** The subroutine to be called must be located in the same task as the LD section called.<br>**Result:** The subroutine call is inserted into the section and a syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  <br>If the subroutine has already been created (see also *Create a Sub-Program Section (SR), page 481*), the procedure ends here.<br>If the subroutine has not yet been created, continue with step 4. |
| 4 | Create the subroutine, see also*Create a Sub-Program Section (SR), page 481*. |

# Property dialog box for coils

## Calling the properties dialog box

see *Displaying the properties, page 734*

## Structure of the properties dialog box

The properties dialog box for **coils**, **negated coils**, **coils for sensing positive/negative transitions** and **set/reset coils** consists of two tabs:

- **General**
  In this tab, you can enter the actual parameter *(see page 754)* of the coil.
- **Comment**
  In this tab, a comment about the coil can be entered.

The properties dialog box for **halt coils** consists of one tab:

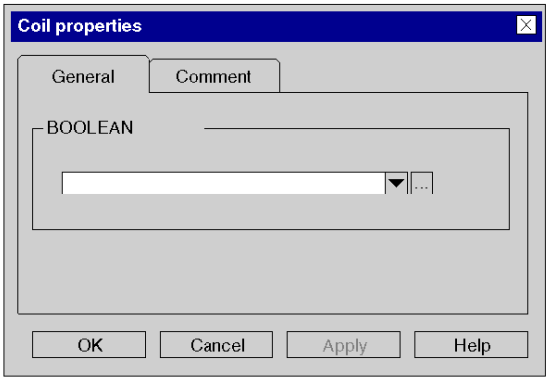
- **Comment**
  In this tab, a comment about the coil can be entered.

The properties dialog box for **call coils** consists of two tabs:

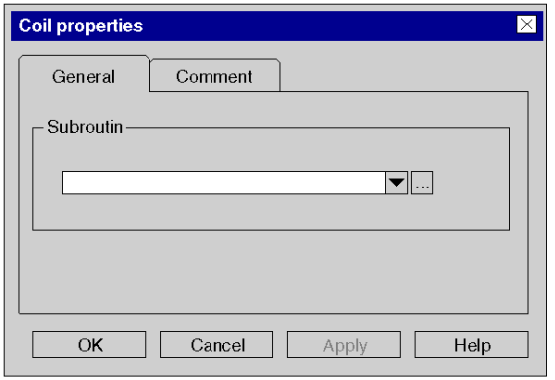
- **General**
  In this tab, you can enter the name of the subroutine to be called *(see page 757)*.
- **Comment**
  In this tab, a comment about the coil/subroutine call can be entered.

## General tab

Representation of the **General** tab:



Representation of the **General** tab for call coils:

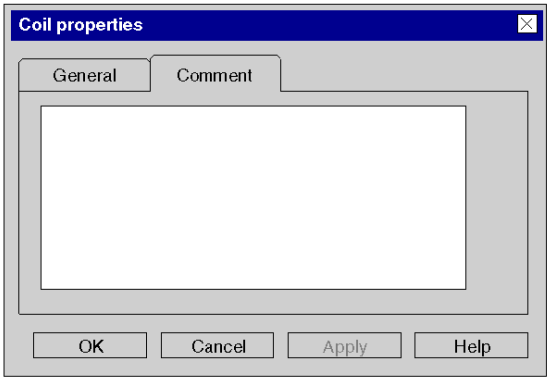



Elements of the **General** tab:

| Element | Description |
|---|---|
| **BOOLEAN Expression** | This text box is only available for **coils**, **negated coils**, **coils for sensing positive/negative transitions** and **set/reset coils**.<br>Enter the name of the actual parameter in this text box.<br>You have the following options:<br>● You can directly enter the name of the variable/address or paste it from the clipboard.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Use the command button **...**to open a variable selection dialog box *(see page 1182)*.<br>The actual parameters which are permitted are:<br>● Boolean variables<br>● Boolean addresses (topological addresses or symbolic addresses) |
| **Subroutine** | This text box is only available for **call coils**.<br>You can enter the name of the subroutine to be called in this text box.<br>You have the following options:<br>● You can directly enter the name of the subroutine or paste it from the clipboard.<br>or<br>● You can select the name of the subroutine from the list of subroutines using the ▼ symbol. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the coil.<br>This comment is displayed as a tooltip when the cursor is placed over the coil. It is independent from the variable comment that can be assigned to the variable in the data editor. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 22.4
## Editing compare blocks

### Overview

This section describes how to edit compare blocks in the LD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Placing compare blocks | 762 |
| Compare block properties dialog box | 766 |

# Placing compare blocks

## Introduction

Compare blocks are only available in the LD programming language. They are used to execute a compare expression (<, >, <=, >=, =, <>) in the ST programming language.

**NOTE:** The same functionality is also possible using ST expressions *(see page 798)*.

If the state of the left link is 1 and the result of the comparison is 1, the state of the right link is 1.

Compare blocks can be placed in any free cell except the last cell directly on the right power rail.

A compare block requires 1 line and 2 columns.

If a compare block is placed in a cell that is occupied by a Boolean link, and the following cell is either empty or also occupied by a Boolean link, then these cells will be occupied by the placed compare block.

If a compare block is placed in a cell, and this cell or the following cell is already occupied by another object, an error message is returned.

A placed compare block automatically creates a connection with its neighboring objects on the left and right if they are of the `BOOL` data type and there are no free cells between them.

## Activating compare block placement mode

You have the following options to activate the compare block placement mode:

- Use the **Edit** → **New** → **Compare Block** menu command.
- Use the **Common Objects** → **Compare Block** menu command from the shortcut menu.
- Press the **Ctrl**+**F7** key combination.
  or
- Choose the 

symbol.

The active compare block placement mode is indicated by the 

cursor symbol.

## Note

In the **Tools** → **Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the checkbox is **cleared**.

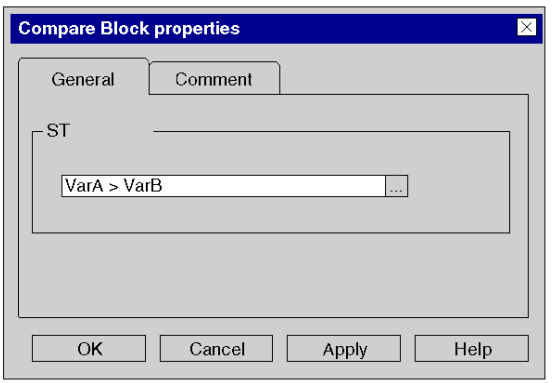
## Placing a compare block

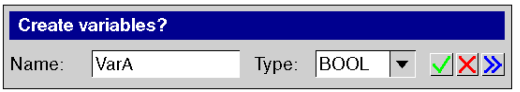
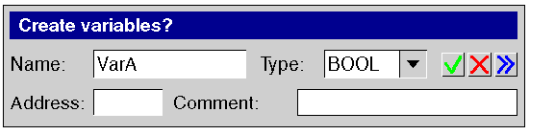
Placing a compare block:

| Step | Action |
|---|---|
| 1 | Activate the compare block placement mode. |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position, and press **Enter**.<br>**Result:**The compare blocks is placed. |
| 3 | Placing other compare blocks:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Defining the compare operation

Defining the compare operation:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 734)* for the compare block.<br>**Result**: The properties dialog box for the compare block is opened, see also *Compare block properties dialog box, page 766*. |
| 2 | Enter the desired compare operation in the text box, see also *General tab, page 766*.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Confirm the compare operation with **OK**.<br>**Result:**<br>● If the selected variables are declared, they are enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>  <br><br>● If the more than one of the selected variables are not declared, they are enabled in the section but not declared. A syntax and semantics check *(see page 717)* is performed. Declare them at this time, using the data editor. See also *Creation of EDT type variable instances, page 370*.<br>● If the **one** of the selected variables is not declared, a variable declaration dialog box opens. In this case, continue with the following steps of this procedure.<br>Variable declaration dialog box<br>  <br><br>**Note:** If you wish to declare the variable later, close the dialog box using the  symbol. In this case, the variable name is enabled in the section, but not declared. |
| 4 | If you do not want to assign the variable an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>Example:<br>  <br> |
| 5 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  <br> |

| Step | Action |
|---|---|
| 6 | Confirm with **Enter** or the ✓ symbol.<br>**Result:**The variable is declared and enabled in the section. A syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  |

# Compare block properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 734*

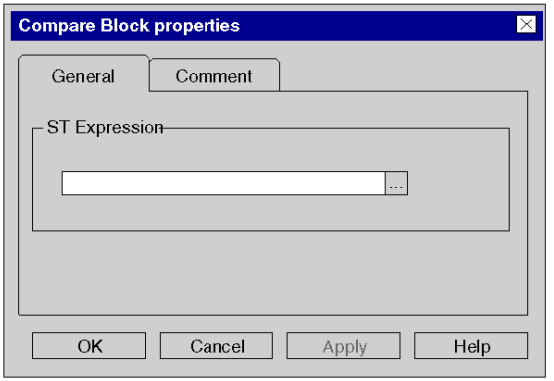
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **General**
  In this tab, you can enter the compare operation *(see page 763)*.
- **Comment**
  In this tab, a comment about the compare block can be entered.

## General tab

Representation of the **General** tab:

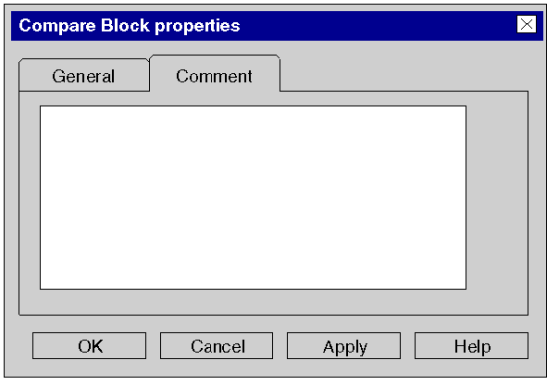


Elements of the **General** tab:

| Element | Description |
|---|---|
| **ST Expression** | Enter the compare operation in this text box, see also *Placing compare blocks, page 762*.<br>You have the following options to variables for the comparison operation:<br>● You can enter the name of the variable/address directly or copy it from the clipboard.<br>● Using the command button **...**to open a variable selection dialog box *(see page 1182)*.<br>A compare block can contain up to 4,096 characters. If not all characters can be displayed, the beginning of the string sequence is followed by an ellipsis (...). |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the**Comment**tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the compare block. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 22.5
## Editing operation blocks

### Overview

This section describes how to edit operate blocks in the LD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Placing Operate Blocks | 769 |
| Operate block properties dialog box | 774 |

# Placing Operate Blocks

## Introduction

Operate blocks are only available in the LD programming language. They are used to execute an ST statement.

When the status of the left link is 1, the ST statement in the block is executed.

All ST statements are allowed **except** the control statements:

- (`RETURN`,
- `JUMP`,
- `IF`,
- `CASE`,
- `FOR`
- etc.)

For operate blocks, the state of the left link is passed to the right link (regardless of the result of the ST statement).

Operate blocks can be placed in any free cell.

A operate block requires 1 line and 4 columns.

If an operate block is placed in a cell that is occupied by a Boolean link, and the following cells are either empty or also occupied by Boolean links, then these cells will be occupied by the placed operate block.

If an operate block is placed in a cell, and this cell or the following 3 cells are already occupied by another object, an error message is returned.

A placed operate block automatically creates a connection with its neighboring objects on the left if they are of the `BOOL` data type and there are no free cells between them.

## Activating operate block placement mode

You have the following options to activate the operate block placement mode:

- Use the **Edit** → **New** → **Operate Block** menu command.
- Use the **Common Objects** → **Operate Block** menu command from the shortcut menu.
- Press the key combination **Alt**+**F7**.
  or
- Choose the 

symbol.

The active operate block placement mode is indicated by the 

cursor symbol.

### Note

In the **Tools** → **Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the checkbox is **cleared**.

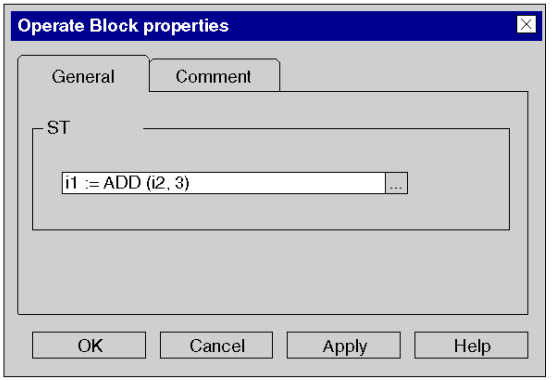
### Placing an operate block

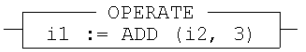
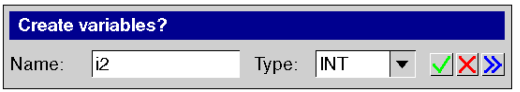
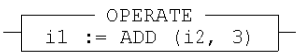
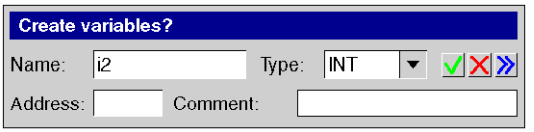
Placing an operate block:

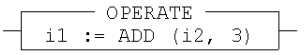
| Step | Action |
|---|---|
| 1 | Activate operate block placement mode. |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:** The operation block is placed. |
| 3 | Placing other operate blocks:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Defining the operation

Defining the operation:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 734)* for the operate block.<br>**Result**: The operate block properties dialog box is opened, see also *Operate block properties dialog box, page 774*. |
| 2 | Enter the desired ST statement in the text box, see also *General tab, page 774*.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Confirm the ST statement with **OK**.<br>**Result:**<br>● If the selected variables are declared, they are enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>  <br>● If the more than one of the selected variables are not declared, they are enabled in the section but not declared. A syntax and semantics check *(see page 717)* is performed. Declare them at this time, using the data editor. See also *Creation of EDT type variable instances, page 370*.<br>● If the **one** of the selected variables is not declared, a variable declaration dialog box opens. In this case, continue with the following steps of this procedure.<br>Variable declaration dialog box<br>  <br>**Note:** If you wish to declare the variable later, close the dialog box using the  symbol. In this case, the variable name is enabled in the section, but not declared. |
| 4 | If you do not want to assign the variable an address or comment, confirm using **Enter** or the  symbol.<br>**Result:**The variable is declared and the ST statement is enabled in the section. A syntax and semantics check *(see page 717)* is performed and the procedure ends here.<br>Example:<br>  |
| 5 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |

| Step | Action |
|---|---|
| 6 | Confirm with **Enter** or the ✓ symbol.<br>**Result:** The variable is declared and the ST statement is enabled in the section.<br>A syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  |

# Operate block properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 734*

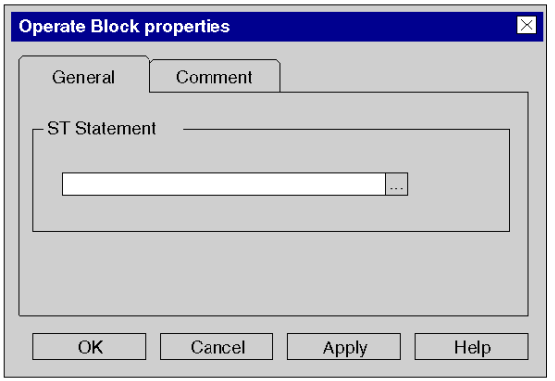
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **General**
  Enter the operation *(see page 771)* in this tab.
- **Comment**
  In this tab, a comment about the operate block can be entered.

## General tab

Representation of the **General** tab:

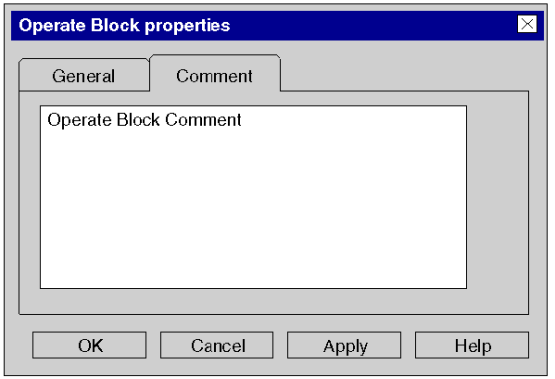



Elements of the **General** tab:

| Element | Description |
|---|---|
| **ST Statement** | Enter the operation in this text box, see also *Placing Operate Blocks, page 769*.<br>You have the following options to select variables for the operation:<br>● You can enter the name of the variable/address directly or copy it from the clipboard.<br>● Using the command button **...**to open a variable selection dialog box *(see page 1182)*.<br>An operate block can contain up to 4,096 characters. If not all characters can be displayed, the beginning of the string sequence is followed by an ellipsis (...). |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment**tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the operate block.<br>This comment is displayed above the operate block if the mixed display mode *(see page 719)* is activated. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 22.6
## Editing FFBs

### Overview

This section describes the various possibilities to edit elementary functions, elementary function blocks, derived function blocks and procedures in the LD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| General information on calling an FFB | 777 |
| Calling an FFB via the FFB input assistant | 779 |
| Calling an FFB via the data selection | 783 |
| Connection of FFBs to the Left Power Rail | 788 |
| Assigning Actual Parameters | 789 |
| Using Public Variables | 801 |
| Expanding Functions | 803 |
| Negating FFB Pins | 805 |
| `EN` and `ENO` Show and Hide | 806 |
| Properties dialog box for elementary functions, procedures and sub-routine blocks. | 807 |
| Properties dialog box for elementary and derived function blocks (FBs) | 809 |

# General information on calling an FFB

## Introduction

FFB is the generic term for:

- Elementary Function (EF) *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- Elementary Function Block (EFB) *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- Derived Function Block (DFB) *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- Procedure *(see Unity Pro, Program Languages and Structure, Reference Manual )* (Procedures are an expansion to IEC 61131-3 and must be enabled explicitly in the **Tools** → **Project Settingsdialog box,** on the **Language Extensions** tab by selecting the **Allow procedures** checkbox).

FFBs can be placed in any free space.

If an FFB is to be placed in an area that is already occupied by an object (contact, coil, other FFB, link), an error message is returned.

## Declaring block instances

Function blocks (elementary and derived) have internal states. If the inputs have the same values, the value on the output can have another value during the individual executions of the function block. For example, with a counter the value on the output is incremented.

The instance name serves as a unique identification for the function block in a project.

Instances can be declared:

- By selecting a function block type and its placement in a section.
- By the declaration of the block instance in the FFB input assistant *(see page 1194)* and its placement in the section.
- By the declaration of the block instance in the Data editor *(see page 359)*.

## Calling FFBs

FFBs can be called in the following ways:

- Via the FFB Input Assistant *(see page 779)* menu command
  - Use the **Edit** → **FFP input assistant... menu command**.
  - Use the menu command **FFB Input Assistant...**in the shortcut menu.
    or
  - Press the **Ctrl**+**I** key combination.
- Via the data selection *(see page 783)*
  - Use the menuc ommand **Edit** → **Data Selection**.
  - Use the menu command**Data selection...** in the shortcut menu.
  - Press the **Ctrl**+**D** key combination.
    or
  - Select the  symbol.
- Via the **Types library browser** per drag & drop
  - Use the **Tools** → **Types Library Manager** menu command.
    or
  - Press the **Alt**+**3** key combination.

# Calling an FFB via the FFB input assistant

## Activating the FFB input assistant

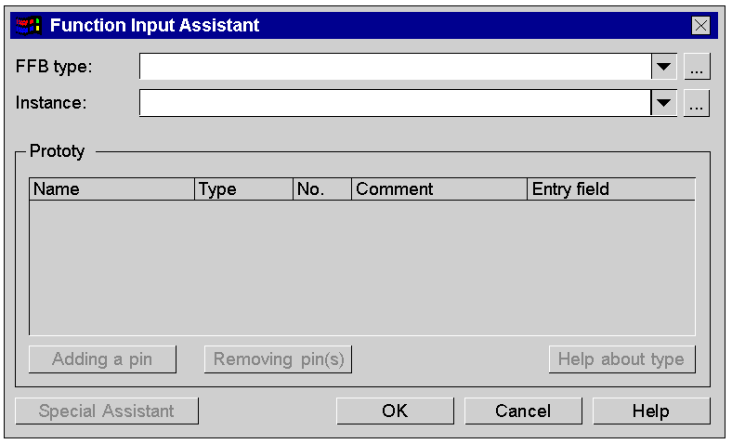
You have the following options to activate the FFB input assistant:

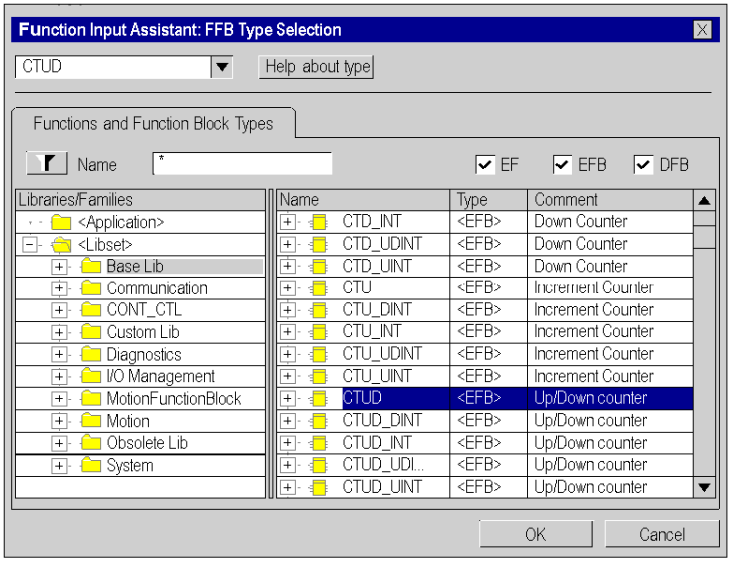
- Use the **Edit → FFB Input Assistant... menu command.** (no object may be selected).
- Use the **menu command** from the shortcut menu (no object may be selected).
  or
- Press the key combination **Ctrl+I** (no object may be selected).

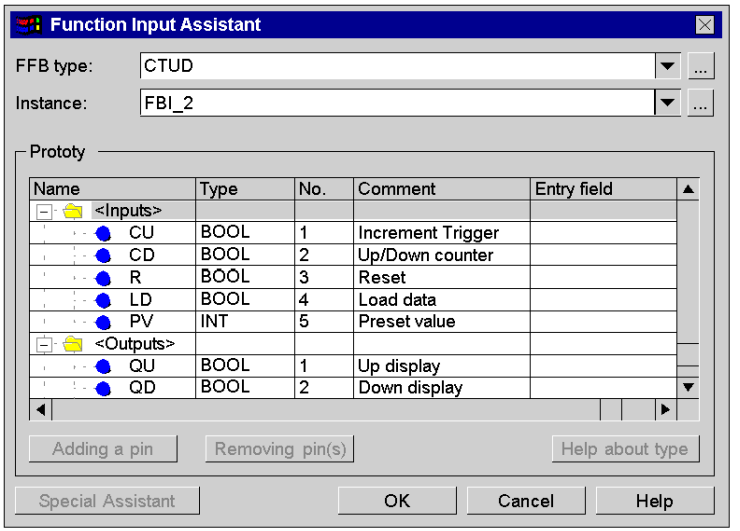
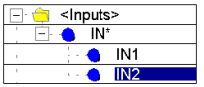
The active FFB placement mode is indicated by the cursor symbol.

## Selecting an FFB type via the Input Assistant

Process the following steps to select an FFB type via the FFB input assistant:

| Step | Action |
|---|---|
| 1 | From the **FFB input assistant....**from the shortcut menu (right-click) for the FFB input assistant *(see page 783)*section.<br>**Result:**<br>The FFB input assistant *(see page 1194)* is opened.<br>  |
| 2 | You have the following options to enter the FFB type with the:**FFB Type** text box:<br>● You can enter the FFB type in the textbox:<br>In this case, continue with step 6 of the procedure.<br>● You can select the FFB type from the list of recently used names using the ▼ symbol.<br>In this case, continue with step 6 of the procedure.<br>● Using the button **...** open the FFB selection dialog *(see page 1187)*. |

| Step | Action |
|---|---|
| 3 | Select the library from the **Libraries/Families** column which contains the desired function block.<br>Tip: If you do not know where to find the FFB you are looking for select the **Libset**, entry to see a list of all the FFBs available independently of their libraries. |
| 4 | Select the desired function block from the **Name** column.<br>Example:<br>  |

| Step | Action |
|---|---|
| 5 | Confirm the selection with **OK**.<br>**Result:** The dialog box is closed and FFB input assistant is activated.<br>Example:<br>  |
| 6 | You have the following options to enter the block instance for function blocks in the **instance** text box:<br>● You can accept the name which is suggested.<br>● You can change the name which is suggested. |
| 7 | The number of inputs *(see page 656)*can be increased with some elementary functions.<br>Expandable pins can be recognized from the following display:<br>  <br>To add further pins select the last pin in the structure and press **Add pin**. |

| Step | Action |
|---|---|
| 8 | To assign the current parameter double-click the **Entry field** cell of the first formal parameter and enter the parameter to be used.<br>You have the following options to enter the variables/addresses:<br>● You can enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the button **...**to open a variable selection dialog box *(see page 1182)*.<br>Assign an actual parameter to all formal parameters of the function block in this way. |
| 9 | Confirm the entries using the **OK** button.<br>**Result:** The FFB input assistant is closed and the FFB input mode is activated. The active FFB placement mode is indicated by the  cursor symbol. |
| 10 | To place the FFB , and click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:** The FFB is inserted and a syntax and semantics check *(see page 717)* is performed. |

# Calling an FFB via the data selection

## Set data selection to active

You have the following options to activate the data exchange:

- Use the **Edit → Data Selection menu command**.
- Use the menu command**Data selection...**from the shortcut menu (right-click) for the section.
- Press the **Ctrl**+**D** key combination.
  or
- Select the  symbol.

**NOTE:** When executing the commands no object can be selected in the section.

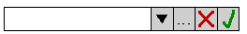
The active FFB placement mode is indicated by the  cursor symbol.

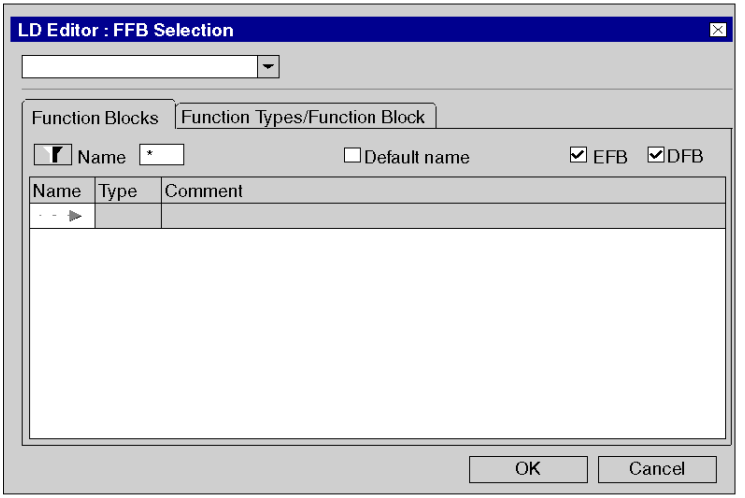
## Note

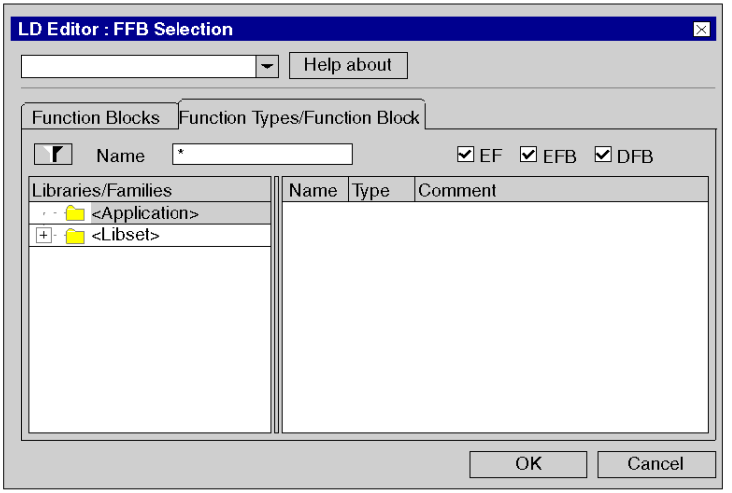
In the **Tools → Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the checkbox is **cleared**.

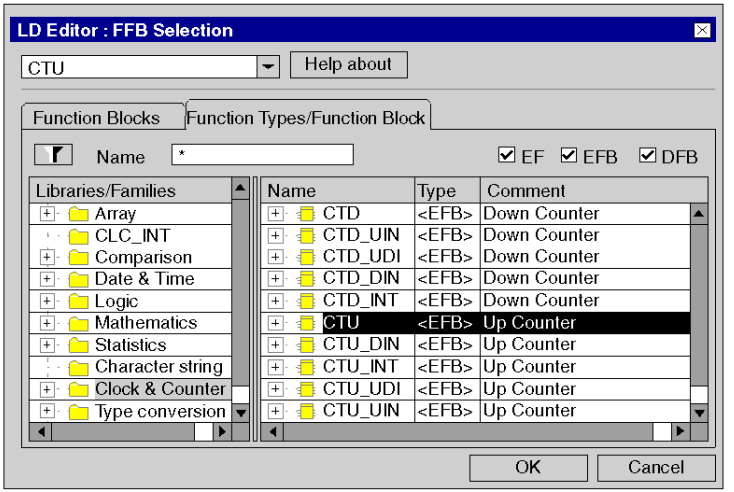
## Selecting an FFB type via the Data Selection

Process the following steps to select an FFB type via the Data selection:

| Step | Action |
|---|---|
| 1 | From the **Data selection....**from the shortcut menu (right-click) for the Data selection *(see page 783)*section.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |

| Step | Action |
|---|---|
| 2 | You have the following options to enter the FFB type:<br>● You can now enter the FFB type and confirm using **Enter**or the ✓button. In this case, continue with step 7 of the procedure.<br>● You can select the name from the list of recently used names using the FFB type▼ symbol and confirm using **Enter**or the button ✓. In this case, continue with step 7 of the procedure.<br>● You can delete your entry by pressing **Esc**or clicking the ✕ button.<br>● Using the button **...you can open** the dialog box for FFB selection *(see page 1184)*.<br>Example of an FFB type selection dialog box:<br>  |

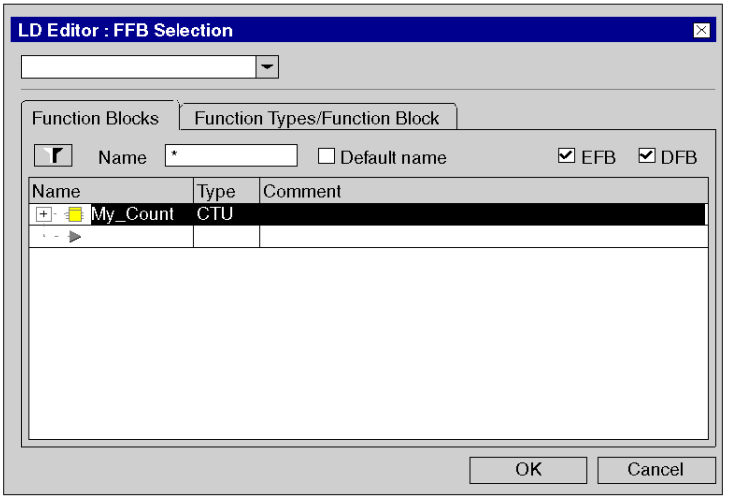
| Step | Action |
|---|---|
| 3 | Select the **Function Types, Function Block Types** tab.<br>**Result:**<br>The tab used for selecting an FFB type *(see page 1185)* is opened.<br>  |
| 4 | Select the library and family from the **Libraries/Families** column which contains the desired function block.<br>**Tip:** If you do not know where to find the FFB you are looking for select the **Libset**, entry to see a list of all the FFBs available independently of their libraries. |

| Step | Action |
|---|---|
| 5 | Select the desired function block from the **Name** column.<br>Example:<br>  |
| 6 | Confirm the selection with **OK**.<br>**Result:** The dialog box is closed and FFB placement mode is activated. |
| 7 | To place the FFB , and click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:** For the selected FFB type, an instance name is automatically generated (only valid for function blocks), the FFB is inserted and placement mode remains active so that further FFBs can be inserted. To end insert mode press **Esc**.<br>**Note:** The automatically generated instance names can be changed to improve clarity, also see *Managing of instances of data belonging to the family of function blocks (EF), page 359*. |

## Selecting a block instance

Process the following steps to select a block instance that has already been declared:

| Step | Action |
|---|---|
| 1 | Activate FFB placement mode.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |
| 2 | Press the button **...**, to open the FFB selection dialog *(see page 1184)*. |

| Step | Action |
|---|---|
| 3 | Select the **Function Blocks** tab (if it is not already active).<br>**Result:**<br>The tab used for selecting a block instance *(see page 1184)* is opened and the desired selection of block instances is displayed.<br>  |
| 4 | Select the desired block instance from the **Name** column. |
| 5 | Confirm the selection with **OK**.<br>**Result:** The dialog box is closed and FFB placement mode is activated. |
| 6 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:** The selected block instance is inserted and the placement mode remains active so that more block instances can be inserted. To end insert mode press **Esc**. |

# Connection of FFBs to the Left Power Rail

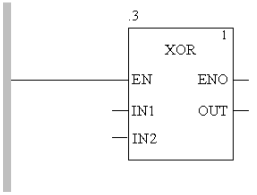
## Introduction

FFBs will only be processed when they are directly or indirectly connected to the left bus bar.

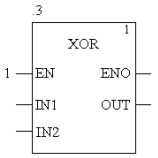
If the FFB should be executed conditionally, the EN input can be pre-linked using contacts or other FFBs, see also section EN and ENO *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the User manual.

## Examples

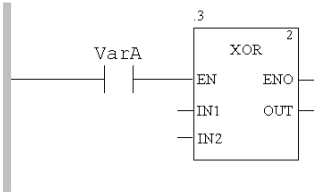
Unconditional FFB call




Unconditional FFB call




Conditional FFB call



# Assigning Actual Parameters

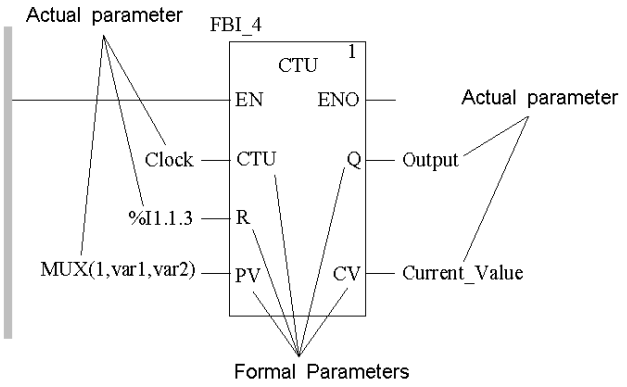
## Introduction

Input and output are required to transfer values to or from an FFB. These are called formal parameters.

Objects are linked to formal parameters; these objects contain the current process states. They are called actual parameters.

The data type of the actual parameter must match the data type of the input/output (formal parameter). The only exceptions are generic inputs/outputs whose data type is determined by the actual parameter. If all actual parameters consist of literals, a suitable data type is selected for the function block.

Formal and Actual Parameters:




Permitted actual parameters are:

- Variables
  - Enter already declared variables *(see page 790)*
  - Enter undeclared variables *(see page 792)*
- Formal parameters of other function blocks *(see page 794)*
- Literals *(see page 797)*
- ST expressions *(see page 798)*
  ST expressions as formal parameters on FBB inputs are an extension of IEC 61131-3 and must be explicitly enabled via the **Tools → Project Settings** dialog box, in the **Language extensions** tab by selecting the **Usage of ST expressions** checkbox.
- Links *(see page 827)* to other LD objects

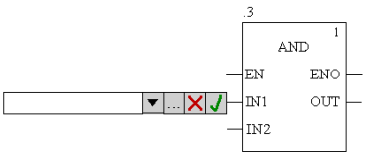
**NOTE:** Please take note of the section Information about Programming *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

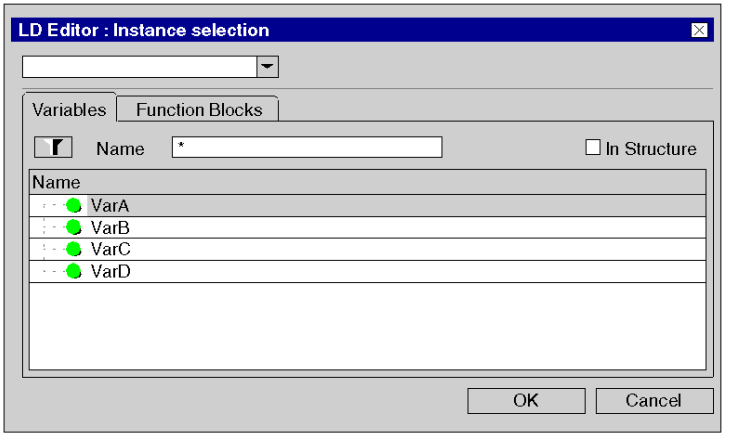
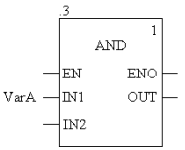
You can make actual parameter assignments for

- each pin individually
  or
- with the function input assistant *(see page 799)* for all pins

## Assign declared variables to a pin

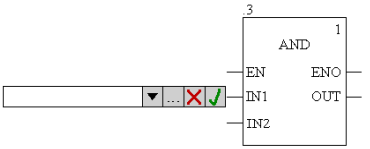
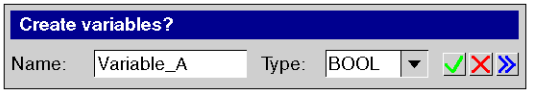
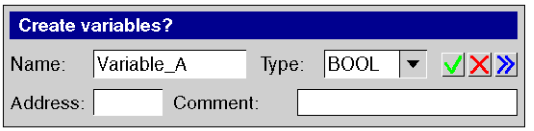
Process the following steps to assign undeclared variables to a pin. (See also *Creation of EDT type variable instances, page 370*.)

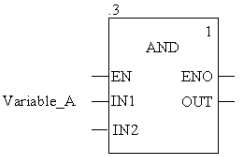
| Step | Action |
|---|---|
| 1 | Open a data selection box for the desired FFB pin by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection...** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |

| Step | Action |
|---|---|
| 2 | You have the following options to enter variables:<br>● You can now enter the variable name and confirm using **Enter**or the ✓button.<br>● You can select the variable names from the list of recently used names using the ▼ symbol and confirm using **Enter** or the ✓button.<br>● You can delete your entry by pressing **Esc** or the ✕ symbol.<br>● Using the button **...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example of a variable selection dialog box:<br>  <br><br>**Result:** The selected variable is inserted and a syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  <br> |

## Assign undeclared variables to a pin

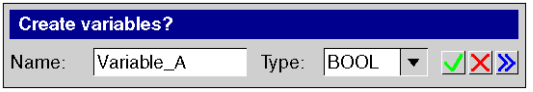
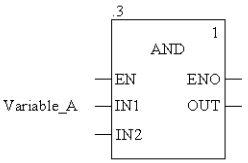
Process the following steps to assign undeclared variables to a pin:

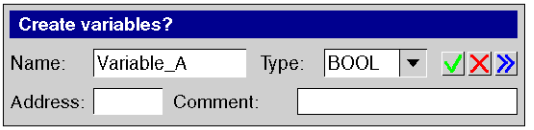
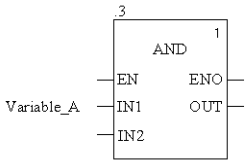
| Step | Action |
|---|---|
| 1 | Open a data selection box for the desired FFB pin by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection...** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>A data selection box *(see page 1911)* opens.<br>  |
| 2 | Enter the variable name and confirm using **Enter**or the  symbol.<br>**Result:** The variable declaration dialog box opens.<br>  |
| 3 | Change the default data type, if desired.<br>**Note:** If you wish to declare the variable later, close the dialog box using the  symbol. In this case, the variable name is enabled on the pin, but not declared. |
| 4 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |

| Step | Action |
|---|---|
| 5 | Confirm with **Enter** or the  symbol.<br>**Result:**The variable is declared and enabled on the selected pin.<br>  |
| 6 | Enter all actual parameters in this way. |

## Declaring variables on a pin

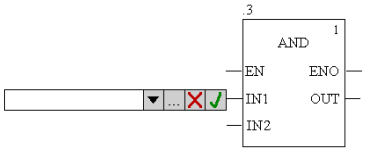
Process the following steps to declare variables on a pin:

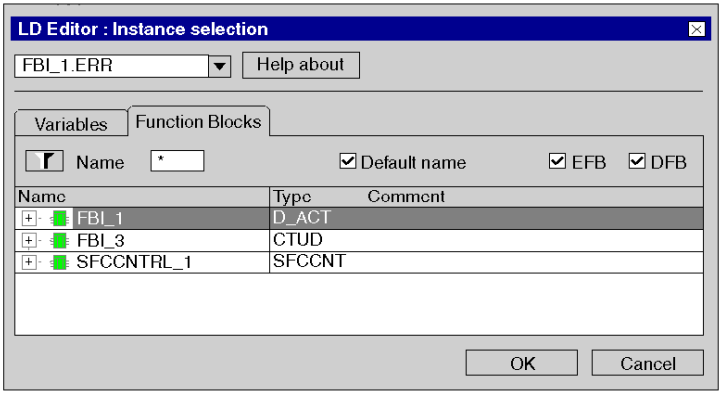
| Step | Action |
|---|---|
| 1 | Open the variable declaration dialog box for the desired variable by.<br>● Select an undeclared variable (red wavy line under the variable name) and the **Create Variable** command from the shortcut menu.<br>● Select an undeclared variable (red wavy line under the variable name) and the **Shift**+**Enter** key combination.<br>**Result:** The variable declaration dialog box opens.<br>  |
| 2 | If you do not want to assign an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears.<br>  |

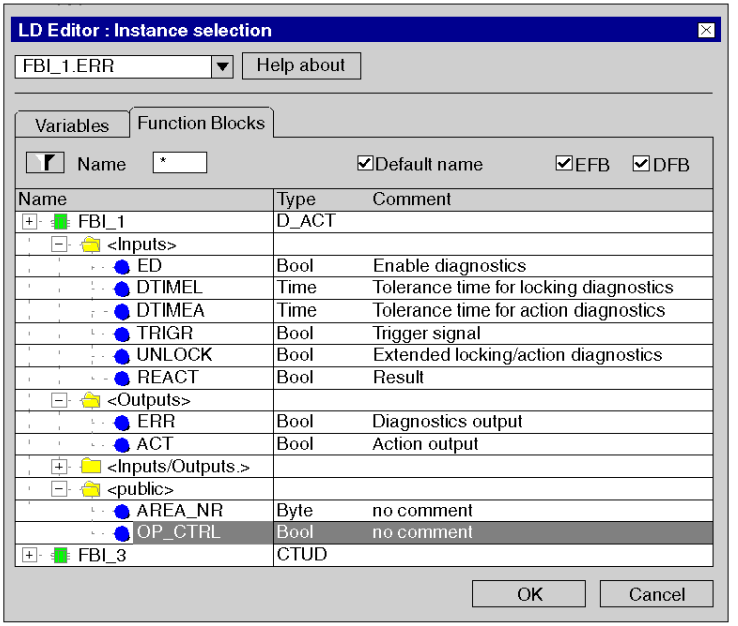
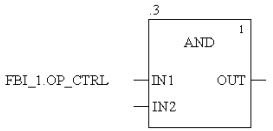
| Step | Action |
|---|---|
| 3 | If you do want to assign the variable an address and/or comment, use the symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 4 | Confirm with **Enter** or the symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears.<br>  |

## Assigning formal parameters to a pin

Process the following steps to assign formal parameters to a pin:

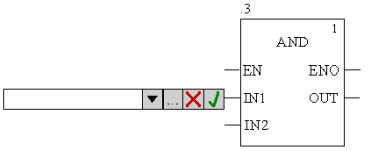
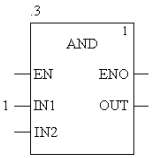
| Step | Action |
|---|---|
| 1 | Select the desired FFB pin. |
| 2 | Open a data selection box by:<br>● double-clicking the pin.<br>● the **Edit** → **Data Selection** menu command.<br>● the **Data Selection** menu command from the shortcut menu.<br>or<br>● the **Ctrl**+**D** key combination.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |

| Step | Action |
|---|---|
| 3 | You have the following options to enter the formal parameter:<br>● You can enter the formal parameter name and confirm using **Enter** or the ✓ button.<br>● You can select the formal parameter name from the list of recently used names using the ▼ symbol and confirm using **Enter** or the ✓ button.<br>● You can delete your entry by pressing **Esc** or clicking the ✗ button.<br>● Using the **button...**to open a variable selection dialog box *(see page 1182)*. |
| 4 | Select the **Function Block** tab.<br>**Result:**<br>The function block instances are displayed.<br>  |

| Step | Action |
|---|---|
| 5 | Expand the display of the inputs, outputs and public variables by clicking their **+** symbols.<br>**Result:**<br>You can now see all available inputs, outputs and public variables of the function block.<br>  <br> |
| 6 | Select the desired formal parameter and confirm with **OK**. |
| 7 | **Result:** The formal parameter selected is inserted and a syntax and semantics check *(see page 616)* is performed.<br>Example:<br>  <br> |

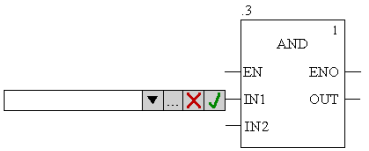
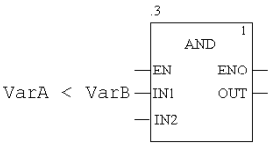
## Assigning literals to a pin

Process the following steps to assign literals to a pin:

| Step | Action |
|---|---|
| 1 | Select the desired FFB pin. |
| 2 | Open a data selection box by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |
| 3 | Enter the literal (e.g. `0`, `1`, `TRUE`, `FALSE`, `3.5`, `t#2ms`) and confirm using **Enter**.<br>**Tips:** If the in the dialog box **Tools** → **Project settings** in the **Language Extensions** tab **Allow leading digits** checkbox is cleared, the following simplifications for entering literals are possible:<br>● When assigning `TIME` literals to a pin of the `TIME` data type, it suffices to enter the numerical value and the unit (such as `h`, `m`, `s`). The prefix (`t#`) is automatically added.<br>● When assigning `REAL` literals to a pin of the `REAL` data type, for integers it suffices to enter the numerical value. The decimal marker (`t#`) is automatically added.<br>**Result:** The literal is inserted and a syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  |

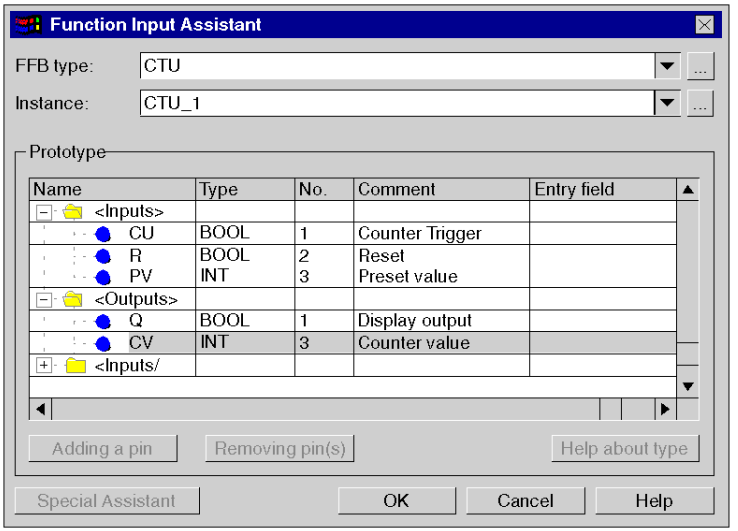
## Assigning ST expressions to a pin

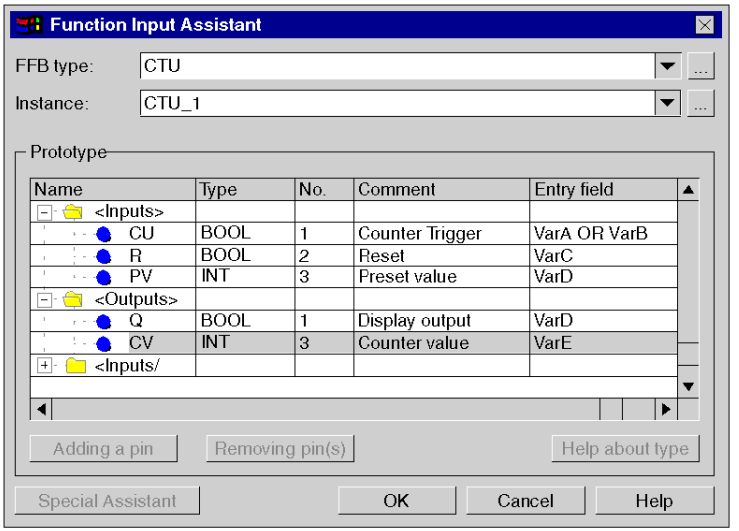
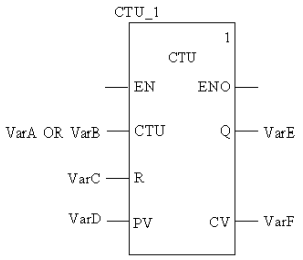
Process the following steps to assign ST expressions to a pin:

| Step | Action |
|---|---|
| 1 | Make sure that the **Usage of ST expressions** checkbox is activated in the **Tools** → **Project Settings** dialog box. |
| 2 | Select the desired FFB pin. |
| 3 | Open a data selection box by:<br>● double-clicking the pin.<br>● selecting the pin and clicking the **Edit** → **Data Selection** menu command.<br>● selecting the pin and clicking the **Data Selection** menu command on the shortcut menu.<br>● selecting the pin and pressing the **Ctrl**+**D** key combination.<br>or<br>● selecting the pin and clicking the  symbol.<br>**Result:**<br>The data selection box *(see page 1911)* opens.<br>  |
| 4 | Enter the ST expression [e.g. `MUX(1,var1,var2)`, `VarA * VarB`, `VarA < VarB`] and confirm using **Enter**.<br>**Result:** The ST expression selected is inserted and a syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  <br>If the variables have already been declared (see also *Creation of EDT type variable instances, page 370*), the procedure ends here.<br>If the variables have not yet been declared, continue with step 5. |
| 5 | Declare all variables used, either:<br>● using the data editor *(see page 370)*<br>or<br>● using **Create variable** in the context menu (possible when only one variable of the ST statement is undeclared). |

## Changing the actual parameter assignment using the function input assistant

Process the following steps to change actual parameter assignment with the function input assistant:

| Step | Action |
|---|---|
| 1 | Select the FFB. |
| 2 | Open the function input assistant *(see page 1194)*:<br>● The **Edit → FFB Input Assistant... menu command.**.<br>● The **FFB Input Assistant... menu command**from the shortcut menu.<br>or<br>● The **Ctrl+I** key combination.<br>**Result:**<br>The function input assistant will be opened.<br>  |

| Step | Action |
|---|---|
| 3 | Double click the **Entry Field** cell of the first formal parameter and enter the actual parameter to be used.<br>You have the following options to enter the variables/addresses:<br>● You can enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the **button...**to open a variable selection dialog box *(see page 1182)*.<br>Assign an actual parameter to all formal parameters of the function block in this way.<br>Example:<br>  |
| 4 | Confirm the entries using the **OK** button.<br>**Result:** The actual parameter is inserted and a syntax and semantics check *(see page 717)* is performed.<br>Example:<br>  |

# Using Public Variables

## Introduction

In addition to inputs/outputs, some function blocks also provide public variables.

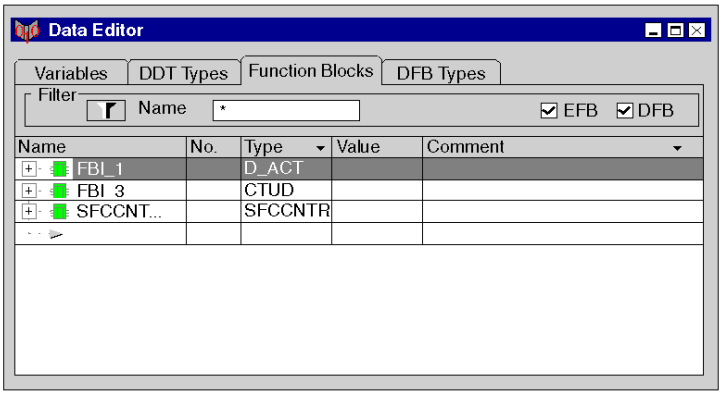
These variables transfer statistical values (values that are not influenced by the process) to the function block. They are used for setting parameters for the function block.

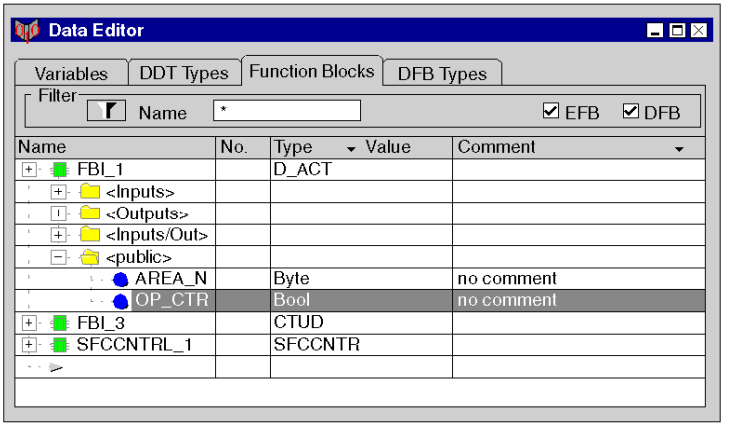
The assignment of values to public variables is made via their initial values.

Public variables are read via the instance name of the function block and the names of the public variables.

## Assigning Values to Public Variables

Carry out the following steps to assign a value to a public variable:

| Step | Action |
|---|---|
| 1 | Place a function block in the section, see also *Calling an FFB via the data selection, page 783*. |
| 2 | Open the data editor, see also *Data Editor Access, page 328*. |
| 3 | Select the **Function Block** tab.<br>**Result:**<br>The function block instances are displayed.<br>  |

| Step | Action |
|---|---|
| 4 | Expand the display of the public variables by clicking the respective **+** symbols.<br>**Result:**<br>You can now see all available public variables of the function block.<br>  |
| 5 | Enter the desired value for the public variable in the **Value** box. |

## Reading public variables

See *Assigning formal parameters to a pin, page 794*

# Expanding Functions

## Introduction

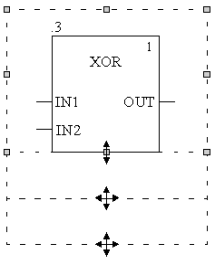
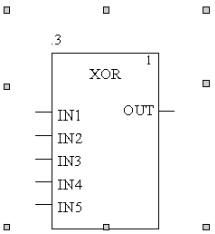
The number of inputs can be increased with some elementary functions.

You find out which functions can be expanded by referring to the descriptions for the individual functions.

**NOTE:** Only expand the function by the number of inputs actually required as non-occupied inputs are assigned a 0 as standard.

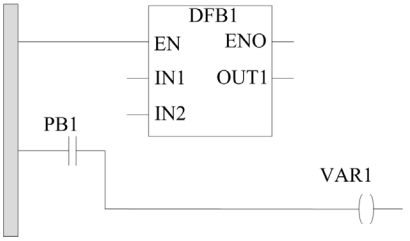
## Expanding a function

Carry out the following steps to expanding a function:

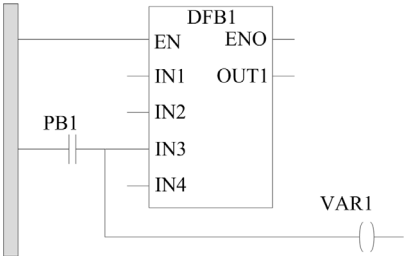
| Step | Action |
|---|---|
| 1 | Select the function. |
| 2 | Position the mouse pointer on the bottom point to be expanded. |
| 3 | Press the left mouse button and drag the function to the required size.<br>Example:<br>  |
| 4 | Release the mouse key.<br>**Result:** The desired number of inputs/outputs have been added.<br>Example:<br>  |

**NOTE:** Expanding a DFB can cause an automatic link with a line placed just under the block.

The following figure shows an example of LD program before a DFB expansion:




The following figure shows the program modification that occurs after a DFB expansion:




Extending the DFB1 will automatically link PB1 to the new input IN3.

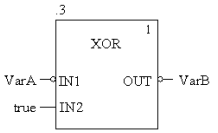
| ⚠ WARNING |
| --- |
| **UNINTENDED PROGRAM EXECUTION**<br>Do not extend a FFB over an other one before adding manually the necessary line.<br>**Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

# Negating FFB Pins

## Introduction

FFB pins (inputs and outputs) of the data type BOOL can be negated.

Negated pins are identified with a circle symbol:




**NOTE:** Negated pins must be connected with a graphical link or a variable. Open negated pins are not allowed.

## Negating an FFB pin

FFB pins can be negated in the following ways:

- Select the pin to be negated and execute the menu command **Negate Pin** in the shortcut menu (right-click).
  or
- Launch the Inversion tool by clicking on the  icon, then click on the pin to be negated.

# EN and ENO Show and Hide

## Introduction

With all FFBs, one EN input and one ENO output can be configured, see also *Unity Pro, Program Languages and Structure, Reference Manual* in the Reference manual.

## EN and ENO show or hide

To show or hide EN and ENO, carry out the following steps:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 734)* for the FFB. |
| 2 | Check the **FFB Properties** check box **Show EN/ENO** to show EN/ENO, or uncheck it to hide EN/ENO.<br>**Note:** If EN or ENO is occupied by an actual parameter or connected with a link, EN/ENO cannot be hidden. |

# Properties dialog box for elementary functions, procedures and sub-routine blocks.

## Calling the properties dialog box

see *Displaying the properties, page 734*

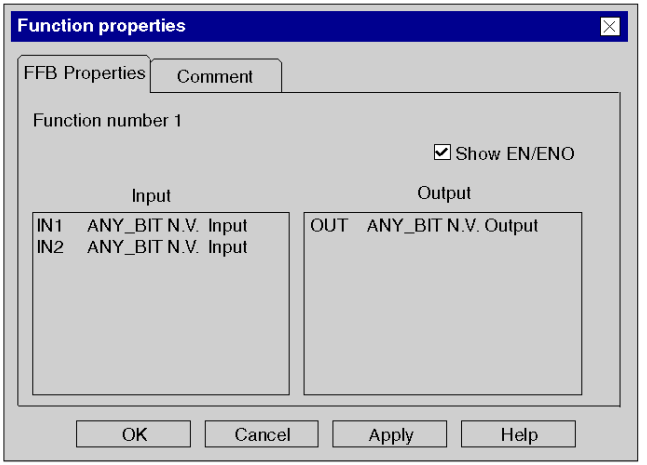
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **FFB Properties**
  General information regarding functions/procedures is displayed in this register.
- **Comment**
  You can enter a comment about the functions/procedures/sub-routine blocks in this tab.

## FFB Properties tab

Representation of the **FFB Properties** tab:



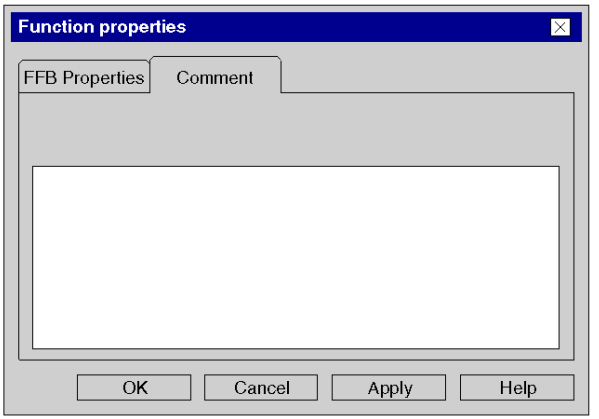

Elements of the **FFB Properties** tab:

| Element | Description |
|---|---|
| **Function** | Specifies the name of the function number and the current execution order. This name cannot be modified. |
| **Show EN/ENO** | When you activate this checkbox the pins EN/ENO are displayed. When you uncheck this checkbox the pins EN/ENO are no longer displayed.<br>**Note:** If EN or ENO is occupied by an actual parameter or connected with a link, the pins EN/ENO cannot be hidden. |

| Element | Description |
|---|---|
| **Input** | Displays the formal parameters, the data types and the actual parameter of the inputs.<br>If the input is not yet linked, the entry **N.L. appears**. |
| **Output** | Displays the formal parameters, the data types and the actual parameter of the outputs.<br>If the output is not yet linked, the entry **N.L. appears**. |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the tab **Comment**:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the FFB.<br>This comment is displayed in the tooltip when the cursor is placed over the function/procedure. |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

# Properties dialog box for elementary and derived function blocks (FBs)

## Calling the properties dialog box

see *Displaying the properties, page 734*

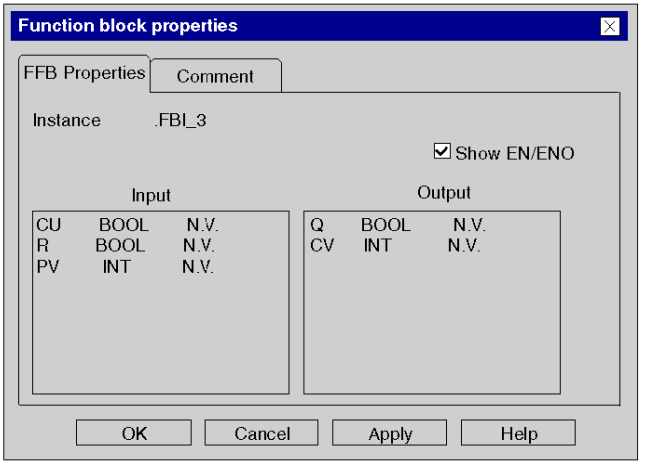
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **FFB Properties**
  General information regarding the FB is displayed in this register.
- **Comment**
  A comment about the FB can be entered in this tab.

## FFB Properties tab

Representation of the **FFB Properties** tab:



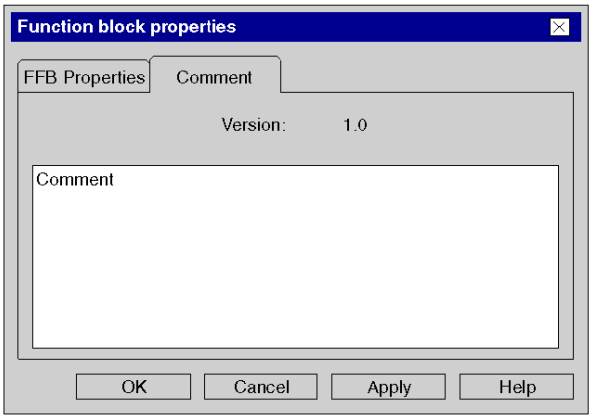

Elements of the **FFB Properties** tab:

| Element | Description |
|---|---|
| **Instance** | Specifies the name of the block instance. You can change this name in the data editor, see also *Managing of instances of data belonging to the family of function blocks (EF), page 359*. |
| **Show EN/ENO** | When you activate this checkbox the pins EN/ENO are displayed. When you uncheck this checkbox the pins EN/ENO are no longer displayed.<br>**Note:** If EN or ENO is occupied by an actual parameter or connected with a link, the pins EN/ENO cannot be hidden. |

| Element | Description |
|---|---|
| **Input** | Displays the formal parameters, the data types and the actual parameter of the FB inputs.<br>If the input is not yet linked, the entry **N.L. appears**. |
| **Output** | Displays the formal parameters, the data types and the actual parameter of the FB outputs.<br>If the output is not yet linked, the entry **N.L. appears**. |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the tab **Comment**:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| **Version:** | Specifies version of the FB code. |
| Text box | Enter a comment about the FB.<br>This comment is displayed in the tooltip when the cursor is placed over the FB. |
| **OK** | Use this button to accept all entries and close the dialog box. |
| **Apply** | Use this button to accept all entries without closing the properties dialog box. |

# Section 22.7
## Returning from a Subroutine or DFB

### Overview

This section describes how to return from a subroutine or DFB to the LD programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Returning from a subroutine or DFB | 812 |
| Return object properties dialog box | 814 |

# Returning from a subroutine or DFB

## Introduction

Every subroutine and every DFB (derived function block) is exited after being processed, i.e. you return to the main program called.

If the subroutine/DFB is left prematurely, the return to the main program can be forced by the return object.

If the status of the left link is ON, a return from the subroutine or DFB to the main program is performed.

Return objects can only be used in subroutines or DFBs. They cannot be used in main programs.

To generate a conditional return, a return object is placed at the end of a series of contacts.

## Selecting the return object

You have the following options to select return objects:

- Use the **Edit** → **New** → **Return** menu command.
- Use the **Common Objects** → **Return** menu command from the shortcut menu.
  or
- Choose the  symbol.

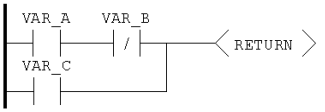
The active placement mode for return objects is indicated by the  cursor symbol.

## Note

In the **Tools** → **Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** check box is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedure explained here is valid for **deactivated** check boxes.

## Placing return objects:

Placing return objects:

| Step | Action |
|---|---|
| 1 | Create a subroutine *(see page 481)* or a DFB *(see page 1255)* in the LD programming language. |
| 2 | Create the logic for the subroutine/DFB. |
| 3 | Select the return object. |
| 4 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:**The return object is inserted.<br>Example:<br>  |
| 5 | To insert further return objects:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

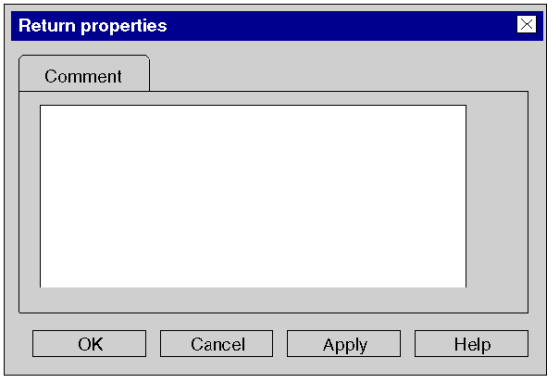
# Return object properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 734*

## Structure of the properties dialog box

Representation of the properties dialog box:




Elements of the properties dialog box:

| Element | Description |
|---|---|
| Text box | Enter a comment about the return object. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 22.8
## Jumps within the Current Section

### Overview

This section describes jumps within the current LD section.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Jumps within the current section | 816 |
| Jump object properties dialog box | 819 |

# Jumps within the current section

## Introduction

Jump objects execute jumps to a jump target *(see page 822)* within the current section.

When the status of the left link is 1, a jump is made to a label (in the current section).

To generate an unconditional jump, the jump object must be placed directly on the left power rail.

To generate a conditional jump, a jump object is placed at the end of a series of contacts.

## Selecting the jump object

You have the following options to select jump objects:

- Use the **Edit** → **New** → **Jump** menu command.
- Use the **Common Objects** → **Jump** menu command from the shortcut menu.

  or

- Choose the  symbol.

The active placement mode for jump objects is indicated by the  cursor symbol.

## Note

In the **Tools** → **Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** check box is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedure explained here is valid for **deactivated** check boxes.

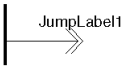
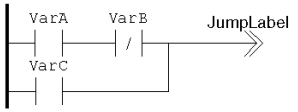
## Placing jump objects:

Placing jump objects:

| Step | Action |
|---|---|
| 1 | Select the jump object. |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position, and press **Enter**.<br>**Result:** The jump object is inserted. |
| 3 | To insert further jump objects:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Defining the jump target

Defining the jump target:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 734)* for the jump object.<br>**Result**: The properties dialog box for the jump object is opened, see also *Jump object properties dialog box, page 819*. |
| 2 | You have the following options to enter the jump target:<br>● You can now enter the name of the jump target and confirm using **Enter**.<br>or<br>● You can select the name of the jump target from the list of available jump targets using the ▾ symbol and confirm using **Enter**.<br>**Result:** The jump target is inserted and a syntax and semantics check *(see page 717)* is performed.<br>Example of an unconditional jump:<br>  <br><br>Example of a conditional jump:<br>  <br><br>If the jump target has already been defined (see also Jump target *(see page 822)*), the procedure ends here.<br>If the jump target has not yet been declared, continue with step 3. |
| 3 | Specify the jump target, see also Jump target *(see page 822)*. |

# Jump object properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 734*

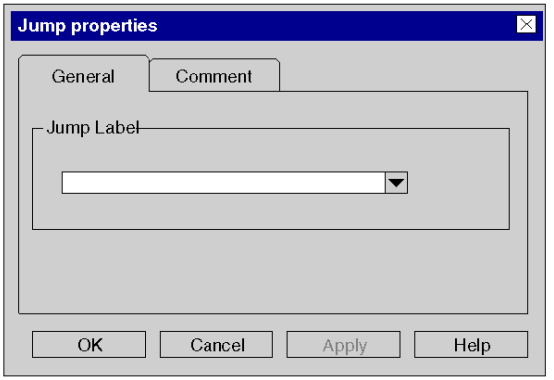
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **General**
  Enter the jump target of the jump *(see page 817)* in this tab.
- **Comment**
  In this tab, a comment about the jump can be entered.

## General tab

Representation of the **General** tab:

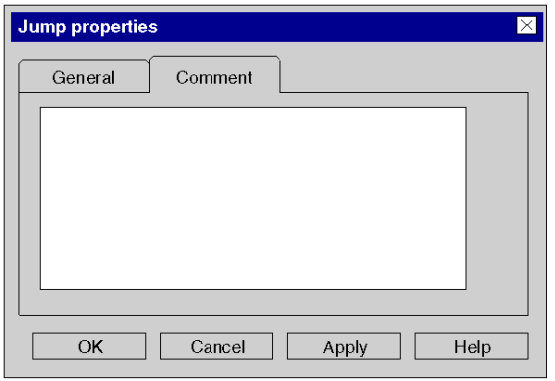



Elements of the **General** tab:

| Element | Description |
|---|---|
| **Jump Label** | Enter the jump target in this text box, see also *Definitions of jump targets (jump labels), page 822*. |
| **OK** | Use this command button to accept all entries and close the properties dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the jump object. |
| **OK** | Use this command button to accept all entries and close the properties dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 22.9
# Definition of the Jump Target (Jump Labels)

## Overview

This section describes how to define jump targets in the LD programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Definitions of jump targets (jump labels) | 822 |
| Jump label properties dialog box | 825 |

# Definitions of jump targets (jump labels)

## Introduction

Jump labels create a jump target for jumps *(see page 816)* within the current section.

Jump labels (jump targets) are represented as text with a colon at the end.

The text is limited to 32 characters and must be unique across the entire section. The text must conform to general naming conventions.

Jump labels can only be placed in the first cell directly on the power rail.

**NOTE:** Jump labels may not "dissect" networks, i.e. an assumed line from the jump label to the right edge of the section may not be crossed by any object. This also applies to Boolean links and FFB links.

To find a specific jump label use the **Gehe zu**dialog box.

## Selecting the jump label

You have the following options to to select jump labels:

- Use the **Edit** → **New** → **Jump Label** menu command.
- Use the **Common Objects** → **Jump Label** menu command from the shortcut menu, or
- Choose the  symbol.

The active placement mode for jump labels is indicated by the  cursor symbol.

## Note

In the **Tools** → **Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedure explained here is valid for **deactivated** checkboxes.

## Placing jump objects

Placing jump objects

| Step | Action |
|---|---|
| 1 | Select the jump label. |
| 2 | Click the target cell on the left power rail.<br>or<br>Use the **arrow keys** to move the gray field to the target position on the left power rail, and press **Enter**.<br>**Result:** The return object is inserted.<br>Example:<br> |
| 3 | Placing other jump label objects:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Position of the Name

Position of the Name

| Step | Action |
|---|---|
| 1 | To enter the name, open the properties dialog box *(see page 734)* for the jump label.<br>**Result**: The jump label properties dialog box is opened, see also *Jump label properties dialog box, page 825*. |
| 2 | You have the following options to enter the name of the jump label:<br>● You can now enter the name and confirm using **Enter**.<br>or<br>● You can select the name of the jump from the list of available jumps using the ▾ symbol and confirm using **Enter**.<br>**Result:** The name is inserted and a syntax and semantics check *(see page 717)* is performed.<br>Example:<br>`JumpLabel1:` |

# Jump label properties dialog box

## Calling the properties dialog box

see *Displaying the properties, page 734*

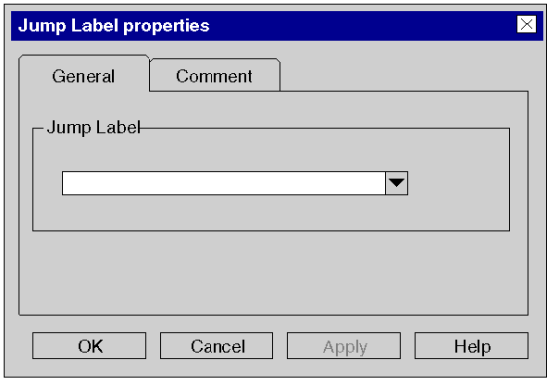
## Structure of the properties dialog box

The properties dialog box consists of two tabs:

- **General**
  Enter the name of the jump label *(see page 823)* in this tab.
- **Comment**
  In this tab, a comment about the jump label can be entered.

## General tab

Representation of the **General** tab:

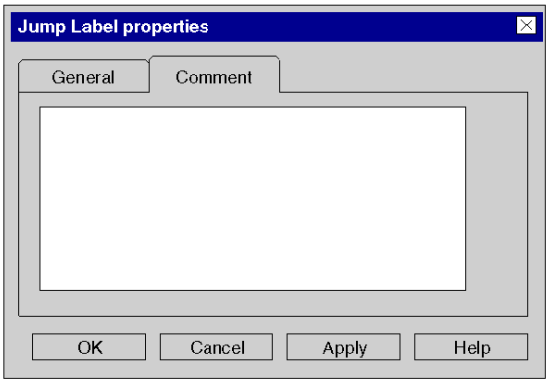



Elements of the **General** tab:

| Element | Description |
|---|---|
| **Jump Label** | Enter the name of the jump label in this list box.<br>The text is limited to 32 characters and must be unique across the entire section. The text must conform to the general naming conventions. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the jump label object. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 22.10
# Editing Links

## Overview

Editing links in the LD programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Select links | 828 |
| Combining Links | 830 |
| Placing links | 832 |
| Editing links | 847 |

# Select links

## Introduction

Links are connections between LD objects (contacts, coils, FFBs etc).

There are two types of links:

- Boolean links
  Boolean links comprise one or more segments that connect Boolean objects (contacts, coils) with one another.
  With Boolean links there are:
  - Horizontal connections
    Horizontal connections enable series switching of contacts and coils.
    The segments of this link can be created individually or as a complex link comprising several segments.
  - Vertical connections
    Vertical connections enable parallel switching of contacts and coils.
- FFB links
  FFB links are a combination of horizontal and vertical segments that connect FFB inputs/outputs with other objects.
  To avoid links crossing each other, FFB links can also be represented in the form of connectors *(see page 588)*.

FFB links are represented with double line thickness to distinguish them from Boolean links.

**NOTE:** A detailed description about links, see also section Links *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

## Selecting links

Selecting links:

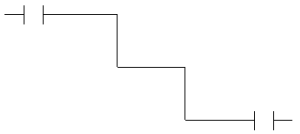
| Type of connection | Call via menu command | Call via symbol | Call via key | Cursor symbol |
|---|---|---|---|---|
| Horizontal connection *(see page 832)* | **Edit → New → Horizontal Connection** or **Link → Horizontal Connection** from the shortcut menu |  | **F7** |  |
| Link tool mode *(see page 833)* (horizontal connection) | - |  | **Alt** + **F6** |  |
| Vertical connection *(see page 834)* | **Edit → New → Vertical Connection** or **Link → Vertical connexion** from the shortcut menu |  | **Shift** + **F7** |  |
| FFB link *(see page 836)* | **Edit → New → Link** |  | **F6** |  |

# Combining Links

## Boolean links

Horizontal and vertical Boolean links can be combined in any way.

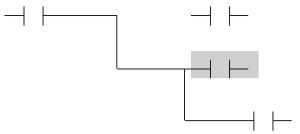
Example:



## Contacts and coils

Contacts and coils link automatically to existing horizontal or vertical links if no free cells are found between them.

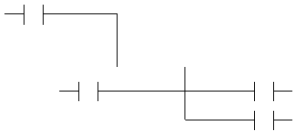
Example:



## Crossing Boolean links

If two Boolean links cross each other they are automatically connected. Since Boolean links may not cross each other there is no special label for them.

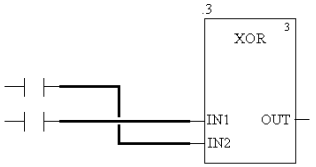
Example:



## Crossing FFB links

If two FFB links cross each other a connection is not made between them. Crossed links are indicated by a "broken" link.

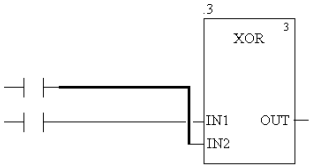
Example:




## Crossing FFB links and Boolean links

If FFB links and Boolean links cross each other, a connection is not made between them. Crossed links are indicated by a "broken" link.

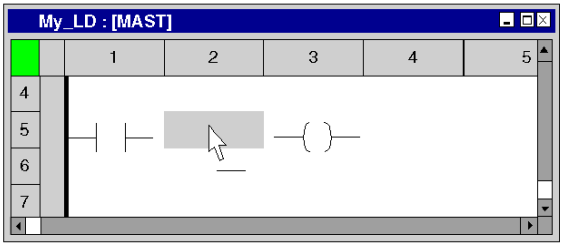
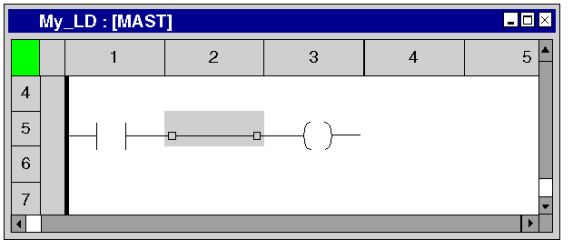
Example:



# Placing links

## Placing horizontal links

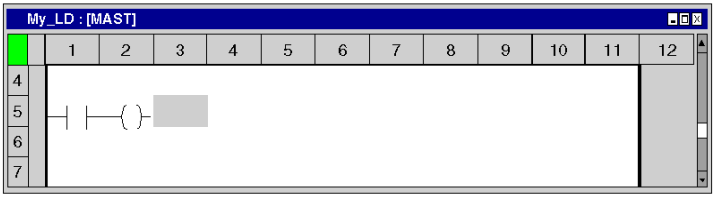
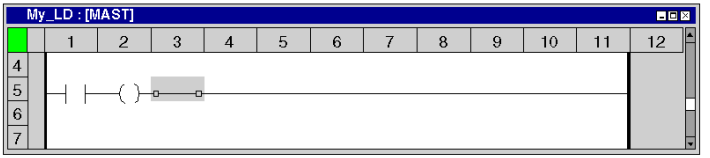
Process the following steps to place a horizontal link:

| Step | Action |
|---|---|
| 1 | Activate the placement mode for horizontal links, see also *Select links*, *page 828*.<br>Example:<br>  |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:** The link is inserted.<br>Example:<br>  <br>See also<br>• *Combining Links*, *page 830*<br>• Placing horizontal and vertical links with FBB links<br>using the mouse *(see page 842)*<br>using the keyboard *(see page 844)*<br>• *Editing links*, *page 847* |

| Step | Action |
|---|---|
| 3 | To place further links of the same type:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Placing horizontal links in link-tool mode

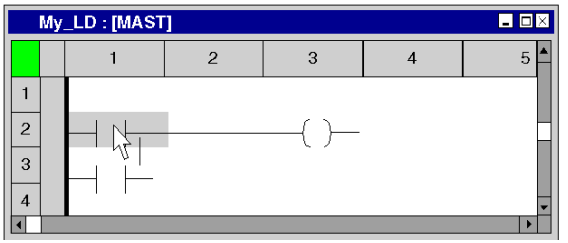
Process the following steps to place a horizontal link in link-tool mode:

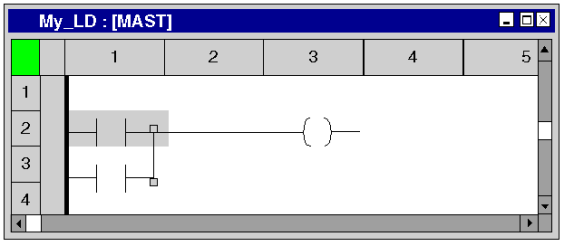
| Step | Action |
|---|---|
| 1 | Activate the link-tool mode, see also *Select links, page 828*.<br>Example:<br>  |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:** The link is inserted and a connection is created to the neighboring Boolean objects to the left and right.<br>If there are no neighboring Boolean objects available, a connection to the power rail is made. This function is a good way to connect coils with the right power rail.<br>Example:<br>  <br>See also<br>● *Combining Links, page 830*<br>● Placing horizontal and vertical links with FBB links<br>using the mouse *(see page 842)*<br>using the keyboard *(see page 844)*<br>● *Editing links, page 847* |

| Step | Action |
|---|---|
| 3 | To place further links of the same type:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Placing vertical links

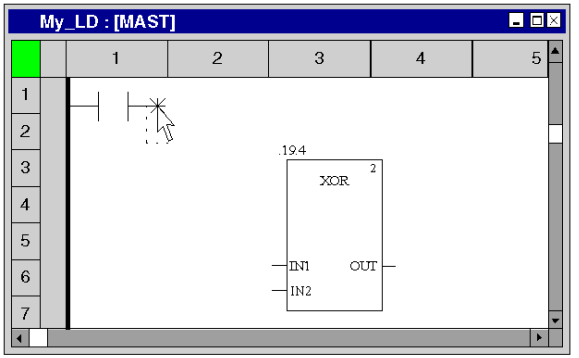
On the right side of the cells, there is space to create a vertical link to the cell below. Process the following steps to place a vertical link:

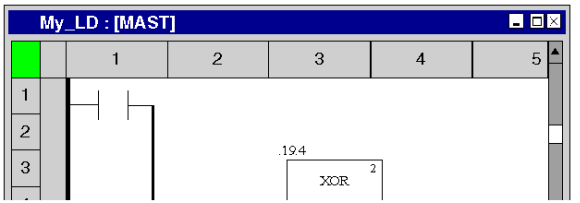
| Step | Action |
|---|---|
| 1 | Activate the placement mode for vertical links, see also *Select links, page 828*.<br>Example:<br>  |

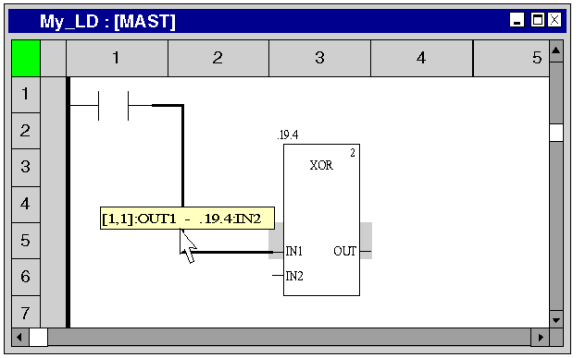
| Step | Action |
|---|---|
| 2 | Click the contact which should be connected with the cell below.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:** The link is inserted and the selection mode is active again.<br>Example:<br>  <br><br>See also<br>● *Combining Links, page 830*<br>● Placing horizontal and vertical links with FBB links<br>using the mouse *(see page 842)*<br>using the keyboard *(see page 844)*<br>● *Editing links, page 847* |
| 3 | To place further links of the same type:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Placing an FFB link with the mouse

Process the following steps to place an FFB link using the mouse:

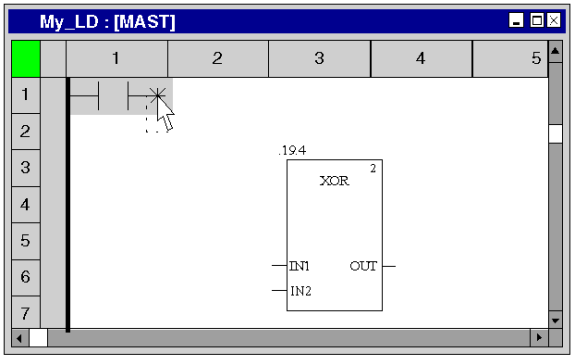
| Step | Action |
|---|---|
| 1 | Activate the placement mode for FFB links. See also *Select links, page 828*. |
| 2 | Place the mouse pointer on the start point of the link.<br>**Notes:** When creating an FFB link, first select the output pin and then the input pin. The cursor symbol indicates the correct position .<br>Example:<br>  |

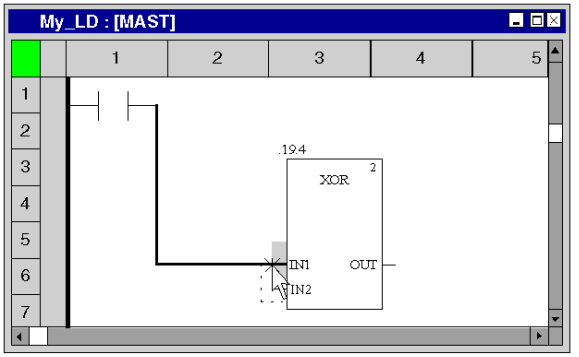
| Step | Action |
|---|---|
| 3 | Click the left mouse button on the start point and drag mouse pointer to the target point of the link.<br>Example:<br>  <br><br>**Note:** For FFB links, at least one FFB input or output must be used. An FFB link between contacts and coils is automatically resolved into a combination of horizontal and vertical links.<br>**Tips:**<br>● By clicking between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● By clicking **Esc**, you quit the mode and the delete the link segments you created.<br>● By clicking **Enter** you quit the mode, and the link segments you created are resolved into a combination of horizontal and vertical links. |

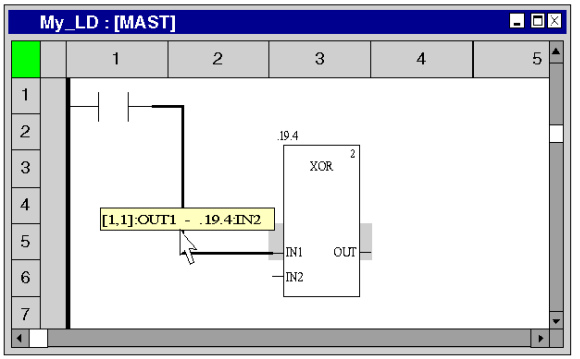
| Step | Action |
|---|---|
| 4 | Click on the target position with the left mouse button<br>**Result:** The link is inserted. When the cursor is placed over an FFB link, the tooltip displays the source and the destination of the link.<br>Example:<br>  <br><br>See also<br>● *Combining Links, page 830*<br>● *Editing links, page 847* |

## Placing an FFB link with the keyboard

Process the following steps to place an FFB link using the keyboard:

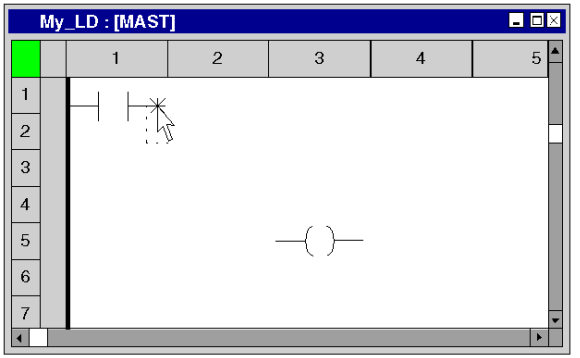
| Step | Action |
|---|---|
| 1 | Use the **arrow keys** to move the gray cell cursor to the source object (for example, contact) of the link.<br>**Note:** When creating an FFB link, first select the output pin and then the input pin. |
| 2 | Activate the placement mode for horizontal links using **F6**. See also *Select links, page 828*. |
| 3 | Press the **Ctrl**+**Enter** key combination.<br>**Result:** The output pin of the object in the gray cell is selected as the start point of the FFB link.<br>Example:<br>  |

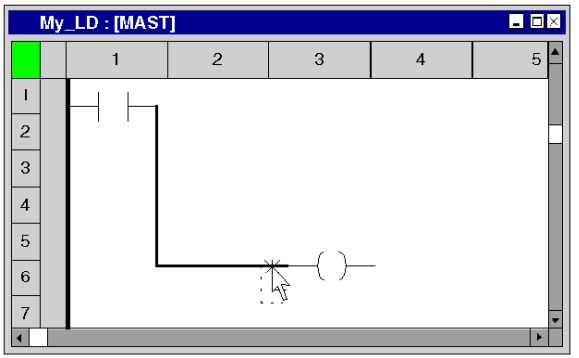
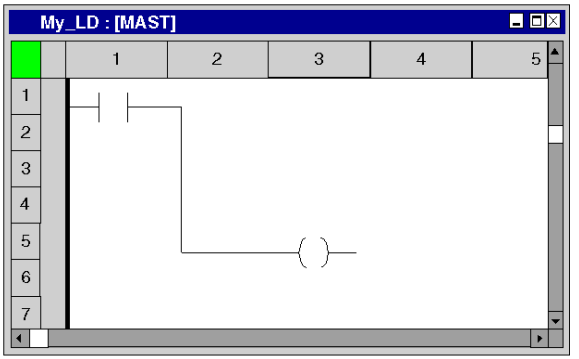
| Step | Action |
|---|---|
| 4 | Use the **Ctrl**+**arrow keys** to move the cursor to the target position of the link.<br>**Tips:**<br>● By pressing the **Ctrl**+**Enter** key combination between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● By clicking **Esc**, you quit the mode and the delete the link segments you created.<br>● By clicking **Enter** you quit the mode, and the link segments you created are resolved into a combination of horizontal and vertical links.<br>Example:<br>  <br><br>**Note:** For FFB links, at least one FFB input or output must be used. An FFB link between contacts and coils is automatically resolved into a combination of horizontal and vertical links. |

| Step | Action |
|---|---|
| 5 | Press the **Ctrl**+**Enter** key combination.<br>**Result:** The link is inserted.<br>**Tip:**When the cursor is placed over an FFB link, the tooltip displays the source and the destination of the link.<br>Example:<br>  <br>See also<br>• *Combining Links, page 830*<br>• *Editing links, page 847* |

## Placing horizontal and vertical links using FFB links with the mouse

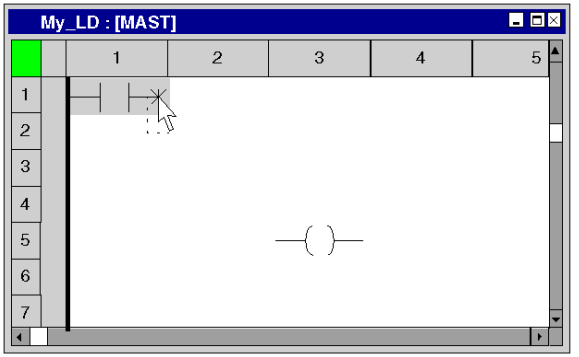
The functions for creating FFB links can also be used to create horizontal and vertical links between other objects (not FFBs). If an FFB link involves no input or output, the FFB link is automatically resolved into a combination of horizontal and vertical links. To do this, process the following steps:

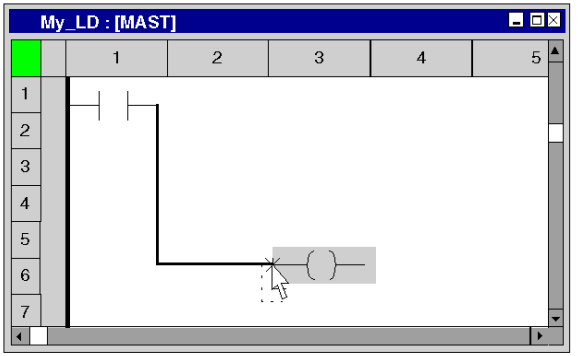
| Step | Action |
|---|---|
| 1 | Activate the placement mode for FFB links. See also *Select links, page 828*. |
| 2 | Place the mouse pointer on the start point of the link.<br>**Notes:** When creating an FFB link, first select the output pin and then the input pin. The cursor symbol indicates the correct position .<br>Example:<br>  |

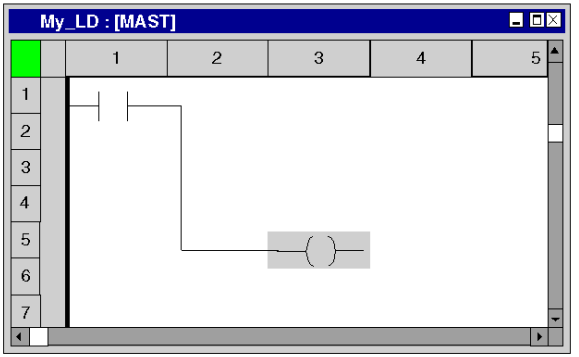
| Step | Action |
|---|---|
| 3 | Click the left mouse button on the start point and drag mouse pointer to the target point of the link.<br>Example:<br>  <br><br>**Tips:**<br>● By clicking between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● By clicking **Esc**, you quit the mode and the delete the link segments you created.<br>● By clicking **Enter** you quit the mode, and the link segments you created are resolved into a combination of horizontal and vertical links. |
| 4 | Click on the target position with the left mouse button<br>**Result:**The link is inserted and the FFB link is automatically resolved into a combination of horizontal and vertical links.<br>Example:<br>  <br><br>See also<br>● *Combining Links, page 830*<br>● *Editing links, page 847* |

## Placing horizontal and vertical links using FFB links with the keyboard

The functions for creating FFB links can also be used to create horizontal and vertical links between other objects (not FFBs). If an FFB link involves no input or output, the FFB link is automatically resolved into a combination of horizontal and vertical links. To do this, process the following steps:

| Step | Action |
|---|---|
| 1 | Use the **arrow keys** to move the gray cell cursor to the source object (for example, contact) of the link.<br>**Note:** When creating an FFB link, first select the output pin and then the input pin. |
| 2 | Activate the placement mode for horizontal links using **F6**. See also *Select links, page 828*. |
| 3 | Press the **Ctrl**+**Enter** key combination.<br>**Result:** The output pin of the object in the gray cell is selected as the start point of the FFB link.<br>Example:<br>  |

| Step | Action |
|---|---|
| 4 | Use the **Ctrl**+**arrow keys** to move the cursor to the target position of the link.<br>**Tips:**<br>● By pressing the **Ctrl**+**Enter** key combination between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● By clicking **Esc**, you quit the mode and the delete the link segments you created.<br>● By clicking **Enter** you quit the mode, and the link segments you created are resolved into a combination of horizontal and vertical links.<br>Example:<br>  |

| Step | Action |
|---|---|
| 5 | Press the **Ctrl**+**Enter** key combination.<br>**Result:** The link is inserted and the FFB link is automatically resolved into a combination of horizontal and vertical links.<br>Example:<br>  <br><br>See also<br>● *Combining Links, page 830*<br>● *Editing links, page 847* |

# Editing links

## Editing Boolean links

Horizontal and vertical Boolean links contain one or more independent segments.

When cutting *(see page 730)*, deleting *(see page 730)*, copying *(see page 731)* and moving *(see page 732)* horizontal and vertical links, the action is executed for the explicitly selected *(see page 727)* segment.

## Editing FFB links

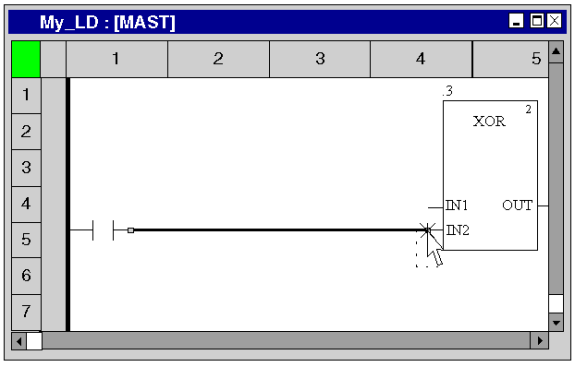
FFB links comprise one or more segments connected to one another.

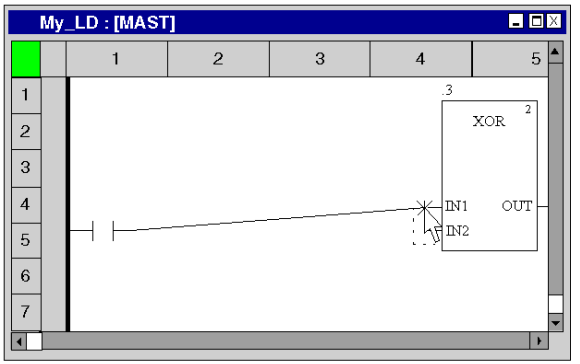
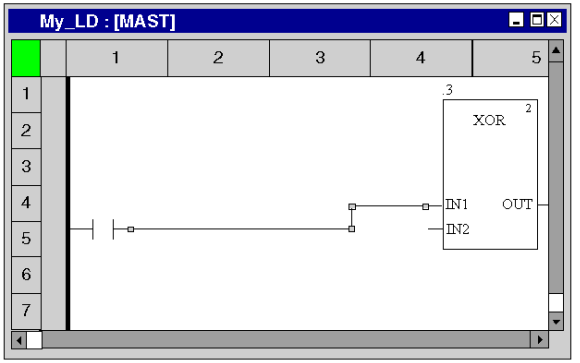
When cutting *(see page 730)*, deleting *(see page 730)*, copying *(see page 731)* and moving *(see page 732)* FFB links, the action is executed for the entire link.

If an FFB link is selected, the target points are displayed at their beginnings, ends and at every direction change

## Modifying connections

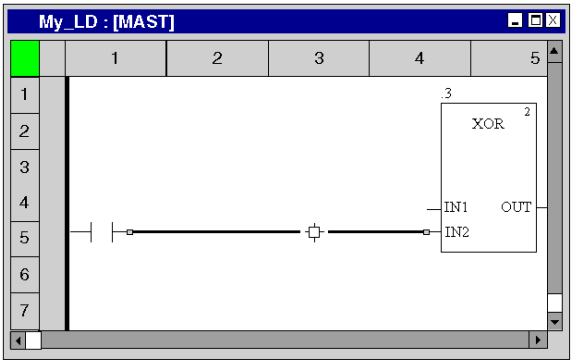
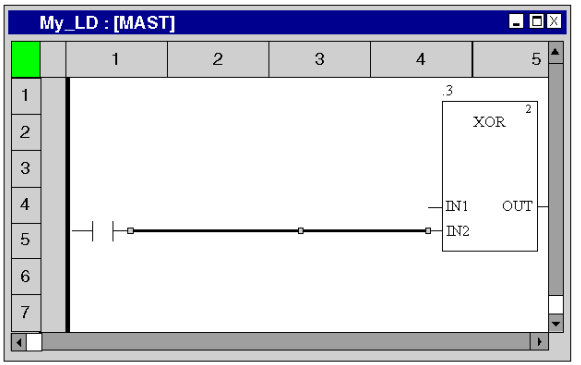
Modifying connections:

| Step | Action |
|---|---|
| 1 | Select the link and place the mouse pointer on the connection to be modified.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | Left-click the connection to be modified and drag the mouse pointer to the new connection.<br>Example:<br>  |
| 3 | Left-click the new target position.<br>**Result:** The link route is changed.<br>Example:<br>  |

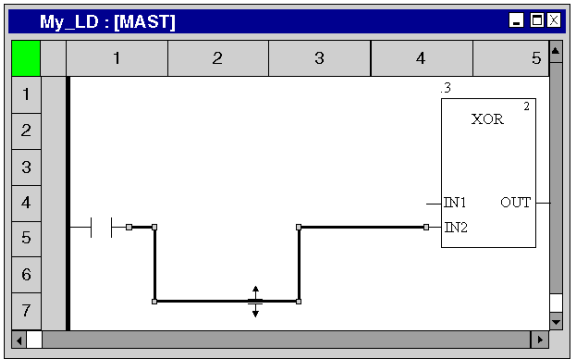
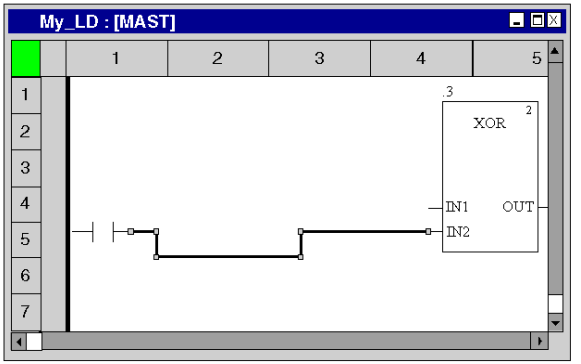
## Creating sizing handles

The creation of angles in FFB links can only be made using sizing handles. If no sizing handles are available they can be created. Process the following steps to create sizing handles in FFB links:

| Step | Action |
|---|---|
| 1 | Select the FFB link. |
| 2 | Press the **Ctrl** key and place the mouse pointer at the target position.<br>**Note:** The correct position is indicated by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Left-click the target position.<br>**Result:**<br>The sizing handle is created.<br>  |

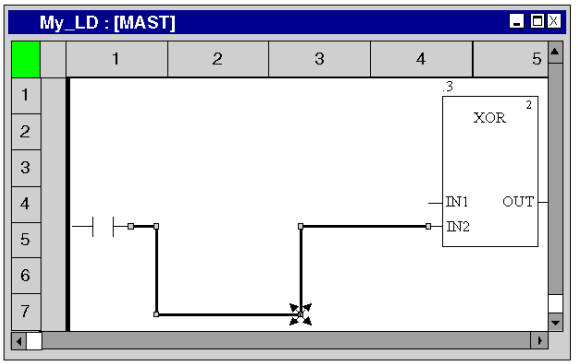
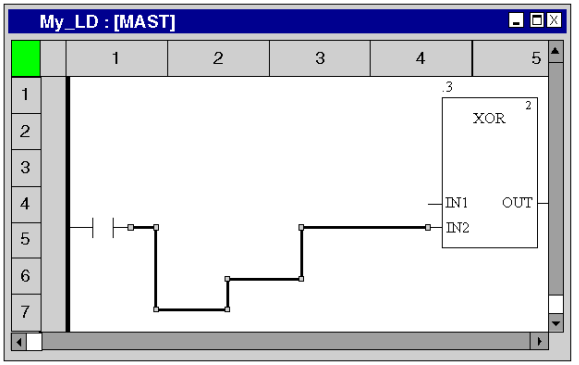
## Moving FFB links vertically/horizontally

Process the following procedures to move an FFB link segment vertically or horizontally:

| Step | Action |
|---|---|
| 1 | Select the FFB link. |
| 2 | Place the mouse pointer on the item to be moved.<br>**Note:** The correct position is indicated by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Click the left mouse button and drag the segment to the target position. |
| 4 | Release the mouse button.<br>**Result:**<br>The change is made.<br>  |

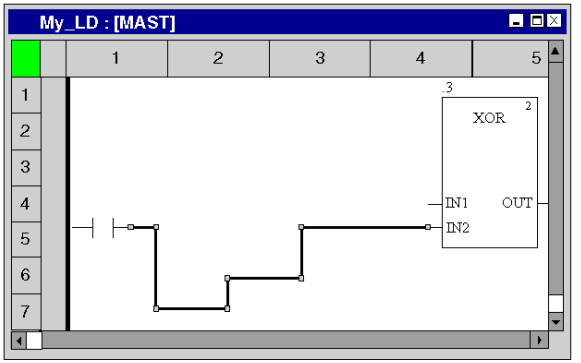
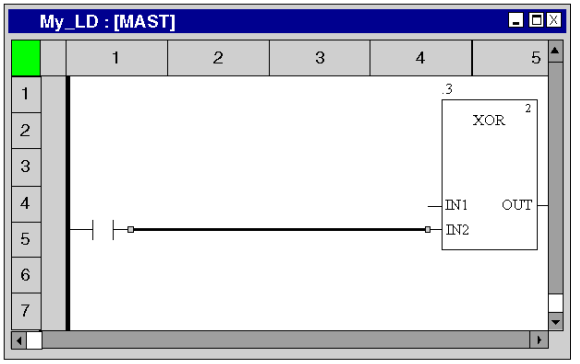
## Creating angles in FFB links

Process the following steps to create angles in FFB links:

| Step | Action |
|---|---|
| 1 | Select the FFB link. |
| 2 | Place the mouse pointer on the source position.<br>**Note:** The correct position is indicated by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Click the left mouse button and drag the source point to the target position. |
| 4 | Release the mouse button.<br>**Result:**<br>The change is made.<br>  |

## Determining the optimum link route

Determining the optimum link route:

| Step | Action |
|---|---|
| 1 | Select the link.<br>Example:<br>  |
| 2 | Use the **Edit** → **Link** → **Route** menu command.<br>or<br>Use the **Route** menu command from the shortcut menu.<br>**Result:** The current link route is changed to the optimum route.<br>Example:<br>  |

# Section 22.11
# Entering Comments

## Entering comments

### Introduction

Comments can be placed in ladder diagram LD in the form of text objects.

Text objects can overlap other objects.

The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill further grid units.

The same rules apply for entering text and navigating within text objects as for editing ASCII text in standard text editors. (To create a line break in text objects you must press the **Ctrl**+**Enter** key combination).

**NOTE:** Keep in mind that each change to a comment (e.g. change to a comment text, change to the size of the text object) makes it necessary to recreate the section involved (**Generation** → **Generate project**).

### Selecting the text object

You have the following options to select text objects:

- Use the **Edit** → **New** → **Comment** menu command.
- Use the **Common objects** → **Comment** menu command from the shortcut menu.
- Press the **F8** key.
  or
- Select the  symbol.

The active placement mode for text objects is indicated by the  cursor symbol.

## Placing text objects:

Placing text objects:

| Step | Action |
|---|---|
| 1 | Select the text object. |
| 2 | Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>**Result:**The text object is inserted.<br>Example:<br>  |
| 3 | Enter the comment. |
| 4 | Confirm the text entered with:<br>● Clicking with the mouse outside of the text object<br>or<br>● Pressing the **Enter** key. |
| 5 | To insert further text objects:<br>● Click the target cell in the LD section.<br>or<br>Use the **arrow keys** to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Section 22.12
# Online functions

## Online Functions

### Online Functions

The description of online functions is found in chapter *Debugging in Ladder Language*, *page 1422*.

# Section 22.13
# LD Reference Data Type

## Reference Data Type in LD

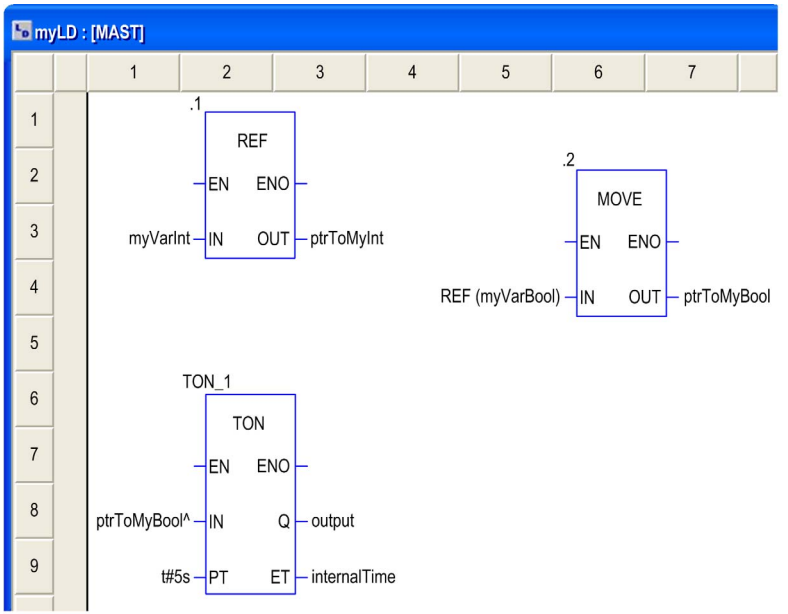
### Introduction

References to variables can be assigned in LD application programs.

A reference can be de-referenced in LD application programs.

For more information, refer to Reference Data Type *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

LB Examples using the Reference Data Type:




The EF, `REF`, takes a variable as an input parameter (type `ANY`) and provides its address as output parameter (type `REF_ANY`).

The EF, `MOVE`, can assign a reference to a variable to another reference. If the input parameter is (`REF(MyVariable)`), `MOVE` has the same effect as the `REF` EF.

The EF, `TON`, takes `ptrToMyBool^` as an input parameter to dereference the reference ptrToMyBool.

# Section 22.14
# Export/Import

## Export/Import LD Sections

### Export/Import

The description for exporting/importing sections is found in the chapter *Import / Export, page 1719*.

# Section 22.15
# Customize LD Editor Colors

## Customize LD Editor Colors

### Overview

To customize LD editor colors, modify the LDColors.ini in the Unity Pro folder.

For more detail on destination folder when Unity Pro is installed, refer to the *Unity Pro Installation Manual (see Unity Pro, Installation Manual)*.

**NOTE:** Make a copy of this file before modifying it.
The LD editor colors are described in the following table.

### Description

This table describes the elements of the color file:

| Parameter | Default value | Description |
|---|---|---|
| `BkAnimation` | `192,192,192` | Editor background color when animation is ON and you are connected in programming mode. |
| `StatementError` | `255,0,0` | Element in error is underlined with this color. |
| `InspectBk` | `0,255,255` | Inspect window background color |
| `InspectText` | `0,0,0` | Inspect window text color |
| `InspectBkMin` | `255,255,0` | Inspect window background color when the value is less than the minimum value defined in the inspect window settings. |
| `InspectBkMax` | `255,0,255` | Inspect window background color when the value is greater than the maximum value defined in the inspect window settings. |
| `BkAnimMonitoring` | `255,211,211` | Editor background color when animation is ON and you are connected in monitoring mode. |
| `AnalyzeError` | `0,0,255` | Drawing color when an element generates an analyze error. |
| `GraphAnimBoolTrue` | `0,150,0` | Color of `ANY_BOOL` variable when its value is TRUE. |
| `GraphAnimBoolFalse` | `255,0,0` | Color of `ANY_BOOL` variable when its value is FALSE. |
| `BkComment` | `255,255,204` | Background color of comment area |
| `BkCommentEdit` | `225,225,225` | Background color of comment area when you are editing it. |
| `BkAnimNumeric` | `255,255,0` | Background color of numeric variables |
| `BlockBreakpoint` | `128,0,0` | Bullet color in the element where the breakpoint is set. |

| Parameter | Default value | Description |
|---|---|---|
| BlockCurrentStep | 255,255,0 | Color of the current element in debug mode |
| AnchorLink | 128,0,0 | Color of anchor link |
| LLFgExpression | 0,0,0 | Text color of boolean expressions |
| LLFgAddress | 100,100,230 | Text color of variable address |
| LLFgSymbol | 0,0,0 | Text color of variable name |
| LLFgComment | 0,128,0 | Text color of variable comment |
| LLBkExpression | 255,255,255 | Background color of boolean expressions |
| LLBkAddress | 255,255,255 | Background color of variable address |
| LLBkSymbol | 255,255,255 | Background color of variable name |
| LLBkComment | 255,255,255 | Background color of variable comment |

# Chapter 23
# SFC Editor

## Overview

This chapter describes the menus and dialogs of the SFC editor.

For a description of the syntax for the SFC programming language, please refer to the Sequential Function Chart SFC *(see Unity Pro, Program Languages and Structure, Reference Manual )* chapter in the Reference manual.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 23.1 | Creating a program with the SFC programming language | 862 |
| 23.2 | General Information about Steps | 897 |
| 23.3 | Editing "normal" steps | 901 |
| 23.4 | Editing macro steps | 915 |
| 23.5 | Editing input steps | 926 |
| 23.6 | Editing Output Steps | 933 |
| 23.7 | Editing actions | 939 |
| 23.8 | Editing Transitions | 951 |
| 23.9 | Editing Jumps | 963 |
| 23.10 | Editing Branches and Joints | 970 |
| 23.11 | Inserting Alternative Strings and Simultaneous Strings | 979 |
| 23.12 | Inserting Step-Transitions and Transition-Step-Sequences | 984 |
| 23.13 | Editing Links | 989 |
| 23.14 | Entering Comments | 1000 |
| 23.15 | Online functions | 1002 |
| 23.16 | Export/Import | 1003 |
| 23.17 | Customize SFC Editor Colors | 1004 |

# Section 23.1
## Creating a program with the SFC programming language

### Overview

This section describes how to create a program in the SFC programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Structure of an SFC Program (Sequential Function Chart) | 863 |
| Creating an SFC Program | 866 |
| Syntax and Semantics Check during Programming | 867 |
| Navigating with the keyboard | 869 |
| Selecting Objects | 870 |
| Undo and Redo Changes | 873 |
| Deleting, cutting, copying, pasting and moving objects | 874 |
| Inserting Rows and Columns | 880 |
| Deleting Rows and Columns | 882 |
| Displaying the object properties | 884 |
| Initialize Search | 885 |
| Initialize Animation Table | 886 |
| Refining steps, macro steps and transitions | 887 |
| Go to | 889 |
| Go to dialog | 891 |
| List and display jumps | 895 |
| Go to Step | 896 |

# Structure of an SFC Program (Sequential Function Chart)

## Introduction

IEC conforming sequential control is created from SFC sections (top level), transition sections and action sections.

SFC sections are only permitted in the projects master task. SFC sections can not be used in other tasks or DFBs.

Each SFC section contains one or more SFC networks (sequences).

## Structure of an SFC section

Zero or more actions belong to every step. A transition condition belongs to every transition.

The last transition of the sequence is always connected to the first step of the sequence (using a graphical link or jump label). Step sequences run cyclically.

## Properties of an SFC program

Properties of an SFC program:

- SFC section always have a grid background.
- Because of performance reasons, it is strongly recommended to create less than 100 SFC sections in a project (makro section are not counted).
- An SFC section contains a maximum of 200 lines and 32 columns.
- SFC objects can theoretically be placed in every unoccupied cell.
- Steps, transitions and jumps each require a cell. Branches and joints do not require their own cell, but are inserted into the respective cell of the step or transition. Text objects can be placed anywhere in the window.
- Steps and transitions are linked with one another through directional links. Two steps can never be directly linked, and must always be separated by a transition, see also section Connection rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
- To prevent step sequences being subdivided, 99 linked steps with the transitions are vertically displayed along with a locking jump with its transition.
- A maximum of 1024 steps can be placed per SFC section (including all their macro sections).
- A maximum of 100 steps can be active per SFC section (including all their macro sections) (multi token), see also **Tools** → **Project Settings** dialog box, **Language extensions** tab, **Sequential Function Chart (SFC)** area.
- A maximum of 64 steps can be set simultaneously per SFC section (multi token), see also **Tools** → **Project Settings** dialog box, **Language extensions** tab, **Sequential Function Chart (SFC)** area.
- A maximum of 100 actions are permitted using the S qualifier per SFC Section.
- A maximum of 20 actions can be assigned to each SFC step.

- The processes of the active signal status take place along the directional links, triggered by the connecting of a transition. The direction of the string process follows the directional links and runs from the under side of the predecessor step to the top side of the successive step. Branches are processed from left to right.
- A syntax and semantics check *(see page 867)* is performed immediately after the statement is entered. The result of this check is displayed in colored text.
- Syntactically or semantically incorrect sections can be saved.

## SFC program objects

The objects of the programming language SFC (Sequential Function Chart) help to divide a section into a number of:

- Step *(see page 901)*
- Macro Step *(see page 915)* (embedded sub-step sequence)
- Transitions *(see page 951)* (transition conditions)
- Jump *(see page 963)*
- Alternative Sequence *(see page 970)*
- Parallel Sequence *(see page 970)*

These objects can be linked with each other using links *(see page 989)*.

Comments for the logic of the program can be added using text objects *(see page 1000)*.

## Edit and view functions

Edit and view functions of the SFC editor:

- Selecting objects *(see page 870)*
- Deleting objects *(see page 874)*
- Cutting *(see page 875)*, copying *(see page 876)* and pasting *(see page 877)* objects
- Moving *(see page 878)* objects
- Undo *(see page 873)* and Redo *(see page 873)*
- Using bookmarks *(see page 576)*
- Finding and replacing *(see page 1235)* variables

## Online functions

Online functions of the SFC editor:

- Setting breakpoints *(see page 1456)*
- Controlling *(see page 1459)* SFC sequences

## SFC save and restore

The SFC_RESTORE function block is used to restart all SFC charts of an application with a given set of active steps at a state, saved before a CPU failure.

In a recovery situation it is possible to set a group of steps that represent a given state in the process and to continue at that point.

(See SFC_RESTORE: SFC Save and Restore *(see Unity Pro, System, Block Library)* in the *System Library*).

# Creating an SFC Program

## Creating an SFC program

Carry out the following procedures to create an SFC program:

| Step | Action |
|---|---|
| 1 | Creating an SFC Section. *(see page 467)* |
| 2 | Place the desired SFC element in the section and define its properties:<br>● Step *(see page 897)*<br>● Macro step *(see page 915)*<br>● Transitions *(see page 951)*<br>● Jumps *(see page 963)*<br>● Alternative strings *(see page 970)*<br>● Simultaneous strings *(see page 970)*<br>● Links *(see page 989)* |
| 3 | Create the last transition of the string and the first step of the string using graphical links or a jump link (cyclical string processing). |

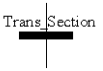
# Syntax and Semantics Check during Programming

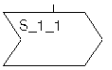
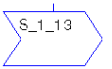
## Introduction

A syntax and semantics check is carried out during program creation. The result of this check is displayed using color objects and text.

## Representation

Representation of colors and labels:

| Color | Description | Example |
|---|---|---|
| Black step symbol | Error free step |  |
| Blue step symbol | Possible errors are:<br>● Input pin not connected<br>● Output pin not connected<br>● Assigned action variable not declared<br>● Assigned `SFCSTEP_TIMES` variable not declared |  |
| Filled yellow step symbol (only Online) | Minimum supervision time *(see Unity Pro, Program Languages and Structure, Reference Manual )* for the step below limit |  |
| Filled magenta step symbol (only Online) | Maximum supervision time *(see Unity Pro, Program Languages and Structure, Reference Manual )* for the step exceeded |  |
| Black transition symbol | Error free transition variable |  |
| Black transition symbol | Error free transition section |  |
| Blue transition symbol | Possible errors are:<br>● Input pin not connected<br>● Output pin not connected |  |
| Red wavy line | Assigned transition variable not declared |  |

| Color | Description | Example |
|---|---|---|
| Black jump symbol | Error free jump |  |
| Blue jump symbol | Possible errors are:<br>● Input pin not connected<br>● Jump target not defined<br>● Invalid jump target |  |

## Quickinfo

If the cursor is placed over faulty text or object, a Quickinfo (Tooltip) with a brief description of the cause of the error appears.

# Navigating with the keyboard

## Using the keyboard

The following keys and key combinations are provided for navigation:

| Key combinations | Motion |
|---|---|
| **Cursor left** | Moves the cursor to the left |
| **Cursor right** | Moves the cursor to the right |
| **Cursor up** | Moves the cursor up |
| **Cursor down** | Moves the cursor down |
| **Ctrl**+**Cursor left** | Moves the cursor one cell left |
| **Ctrl**+**Cursor right** | Moves the cursor one cell right |
| **Ctrl**+**Cursor up** | Moves the cursor one cell up |
| **Ctrl**+**Cursor down** | Moves the cursor one cell down |
| **Home** | Moves the cursor to the first displayed pixel of the current line |
| **End** | Moves the cursor to the last displayed pixel of the current line |
| **Ctrl**+**Home** | Shows the top left cell of the section (the position of the cursor is not affected by this function) |
| **Ctrl**+**End** | Shows the bottom right cell of the section (the position of the cursor is not affected by this function) |
| **Page Up** | Scrolls one page up (the position of the cursor is not affected by this function) |
| **Page Down** | Scrolls one page down (the position of the cursor is not affected by this function) |
| **Ctrl**+**Page Up** | Scrolls one page left (the position of the cursor is not affected by this function) |
| **Ctrl**+**Page Down** | Scrolls one page to the right (the position of the cursor is not affected by this function) |
| **Ctrl**+**Alt**+**Page Up** | Displays the previous section (including via the **View** → **Previous Section**menu). |
| **Ctrl**+**Alt**+**Page Down** | Displays the next section (including via the **View** → **Next Section**menu). |
| **Spacebar** | Selects the objects in the cell where the cursor is currently positioned. |
| **Alt**+**Enter** | Opens the properties dialog box for the selected object. |

# Selecting Objects

## Select and Placement Mode

Selecting objects occurs in select mode.

Select mode can be activated by:

- Using the **Edit** → **Select Mode** menu command
- Using the  symbol
  or
- Pressing the **Esc** key

The  cursor symbol indicates that select mode is active.

## Selecting an Object

| Using the mouse | Using the keyboard |
|---|---|
| Left-click the object you want to select. | 1. Move the cursor to the object to be selected, see also *Navigating with the keyboard, page 869*.<br>2. Press the **spacebar**. |

## Selecting Several Objects

| Using the mouse | Using the keyboard |
|---|---|
| 1. Click the left mouse button and keep it pressed.<br>2. Drag the mouse across the objects you want to select.<br>or<br>1. Left-click the first object you want to select.<br>2. Press the **Ctrl** key and keep it pressed.<br>3. Left-click the next object you want to select.<br>4. Repeat these steps until the desired objects are selected. | 1. Move the cursor to the first object to be selected, see also *Navigating with the keyboard, page 869*.<br>2. Press the **spacebar**.<br>3. Move the cursor to the next object to be selected<br>4. Press the **Ctrl**+**spacebar** key combination.<br>5. Repeat these steps until the desired objects are selected. |

## Selecting by Rows

| Using the mouse | Using the keyboard |
|---|---|
| Selecting the contents of a row:<br>**1.** Left-click the number of the row you want to select in the vertical ruler.<br>Selecting the contents of several rows:<br>**1.** Left-click the number of the first row you want to select in the vertical ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last row you want to select in the vertical ruler.<br>**Note:** Pressing the **Shift** key makes it possible to combine this procedure with the procedure for selecting by columns. | - |

## Selecting by Columns

| Using the mouse | Using the keyboard |
|---|---|
| Selecting the contents of a column:<br>**1.** Left-click the number of the column you want to select in the vertical ruler.<br>Selecting the contents of several columns:<br>**1.** Left-click the number of the first column you want to select in the vertical ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last column you want to select in the vertical ruler.<br>**Note:** Pressing the **Shift** key makes it possible to combine this procedure with the procedure for selecting by rows. | - |

## Selecting All (Entire Contents of the Section)

| Using the mouse | Using the keyboard |
|---|---|
| Use the **Edit** → **Select All** menu command. | Press the **Ctrl**+**A** key combination. |

## Deselecting Objects

| Using the mouse | Using the keyboard |
|---|---|
| Left-click an empty space in the section. | **1.** Move the cursor to an empty space in the section, see also *Navigating with the keyboard, page 869*.<br>**2.** Press the **spacebar**. |

# Undo and Redo Changes

## Undo changes

Undo the last changes in the following ways:

- Use the **Edit** → **Undo** menu command.
- Press the **Ctrl**+**Z** key combination.
  or
- Select the  symbol.

Undo one change each time the function is carried out.

The function can be executed a maximum of 10 times.

If steps or transitions have been deleted, "Undo" means that the step and transition are restored, including the information assigned to them.

If a deleted step/deleted transition has an action/transition section assigned to it, only the reference to the action/transition section is restored and not the action/transition section itself.

## Redo changes

Using **Redo**, you can redo changes that have been made:

- Use the menu command **Edit** → **Redo**.
- Press the **Ctrl**+**Y** key combination.
  or
- Select the  symbol.

Redo one undone action each time the function is carried out.

The function can be executed a maximum of 10 times.

## Limits

After any of the following actions has been carried out**Undo** and **Redo** cannot be used.

- Deleting a step or a macro step in online mode.
- Toggling between **Normal View** and **Advanced View** and vice versa.

# Deleting, cutting, copying, pasting and moving objects

## Deleting objects

Deleting objects:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 870)* the object to be deleted.<br>2. Use the **Edit** → **Delete** menu command. | 1. Select *(see page 870)* the object to be deleted.<br>2. Press the **Entf** key. |
| **Result:**<br>● The selected object (and its graphical link when available) is deleted from the section.<br>● If action sections are assigned to a step, (in offline mode) you are asked to confirm whether the action sections should be deleted. (In Online mode no confirmation is sought and the action sections remain until they are explicitly deleted.)<br>● If a transition section is assigned to a transition, (in offline mode) you are asked to confirm whether this transition section should also be deleted. (In online mode no confirmation is sought and the transition section remains until it is explicitly deleted.)<br>● (If a transition variable is assigned in transition it remains until it is explicitly deleted.)<br>● If steps or transitions are to be deleted with step variables that will be used in the program (including on the operating screen or in animation tables),<br>❍ a message will appear in the online mode that the step variables are used in the program and can only be deleted in the offline mode. If the question whether the operation should be continued is confirmed with **Yes**, the mode will change to offline and the object will be deleted.. **No** will terminate the operation.<br>❍ a message will appear in the offline mode that the step variables are used in the program. If the question whether the operation should be continued is confirmed with **Yes**, the object will be deleted.. With**No** the operation will be terminated.<br>● When deleting a macro step, the associated macro step section is moved to the **Unused macros** directory for the current SFC section.<br>● If a step or macro step is deleted in online mode it cannot be undone using **Undo**. | |

**NOTE:** To delete entire rows or columns from the section, select the lines or columns in the line box, see also *Deleting Rows and Columns, page 882*.

## Cutting objects

Cutting objects:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 870)* the object to be cut.<br>2. Use the **Edit** → **Cut** menu command.<br>or<br>Use the menu command **Cut** in the shortcut menu (right mouse button).<br>or<br>Click the  symbol. | 1. Select *(see page 870)* the object to be cut.<br>2. Press the **Ctrl**+**X** key combination. |
| **Result:**<br>● The selected object (and its graphical link when available) is deleted from the section.<br>● If action sections are assigned to a step, (in offline mode) you are asked to confirm whether the action sections should be deleted. (In Online mode no confirmation is sought and the action section remains until it is explicitly deleted.)<br>● If a transition section is assigned to a transition, (in offline mode) you are asked to confirm whether this transition section should also be deleted. (In online mode no confirmation is sought and the transition section remains until it is explicitly deleted.)<br>● (If a transition variable is assigned in transition it remains until it is explicitly deleted.)<br>● If steps or transitions are to be deleted with step variables that will be used in the program (including on the operating screen or in animation tables),<br>❍ a message will appear in the offline mode that the step variables are used in the program. If the question whether the operation should be continued is confirmed with **Yes**, the object will be deleted.. With**No** the operation will be terminated.<br>❍ an additional message is displayed in the online mode that **Create project** can only be performed in the offline mode.<br>● When deleting a macro step, the associated macro step section is moved to the **Unused macros** directory for the current SFC section.<br>● If a step or macro step is deleted in online mode it cannot be undone using **Undo**. | |

The cut object can be inserted *(see page 877)* in any other position (also in another SFC section).

## Copying objects to the clipboard

Copying objects to the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 870)* the object to be copied.<br>**2.** Use the **Edit** → **Copy** menu command.<br>or<br>Use the **Copy** menu command from the shortcut menu (right-click).<br>or<br>Click the  symbol. | **1.** Select *(see page 870)* the object to be copied.<br>**2.** Press the **Ctrl**+**C** key combination. |
| **Result:**<br>● The selected object and associated information is copied to the clipboard.<br>● Graphical links can only be copied if they and their partner objects are explicitly selected.<br>● If action sections are assigned to a step, only the reference to the action section is copied to the clipboard, not the action section itself.<br>● If transition sections are assigned to a transition, only the reference to the transition section is copied to the clipboard, not the transition section itself.<br>● When copying macro steps, only the macro step is copied and not its macro step section. | |

The copied object can be inserted *(see page 877)* in any other position (also in another SFC section).

## Pasting objects from the clipboard

Pasting objects from the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Use the menu command **Edit** → **Insert**.<br>or<br>Use the menu command **Paste** in the shortcut menu (right mouse button).<br>or<br>Click the  symbol.<br>**2.** Left-click the target position. | **1.** Press the **Ctrl**+**V** key combination.<br>**2.** Move the cursor to the target position using the arrow keys.<br>**3.** Press **Enter**. |
| **Result:**<br>● The objects contained in the clipboard and their associated information are inserted into the section.<br>● Graphical links can only be inserted if they and their partner objects have been explicitly copied to the clipboard.<br>● If the objects to be inserted are placed on a graphical link, the link is undone and the objects are inserted into the structure.<br>● When inserting steps and macro steps they are automatically assigned a unique name.<br>● If an action section is assigned to a step, only the reference to the action section is inserted into the section together with the step and not the action section itself.<br>If an action section is not declared in the current section, the action with the action section is deleted when it is inserted.<br>● If a transition section is assigned to a transition, only the reference to the transition section is inserted into the section together with the transition and not the transition section itself.<br>A transition section may only be referenced once in a project. That means if a reference is already available for a transition during insertion, only the transition object is inserted (without a reference).<br>● When inserting macro steps, only the macro step is inserted and not its macro step section. A new macro step section is created instead. The names of the steps within this macro step section are automatically matched to the (new) macro step names. | |

## Moving objects (Drag & Drop)

Moving objects (Drag & Drop):

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 870)* the object to be moved.<br>**2.** Position the mouse pointer on the selected object. (on one of the selected objects if several are selected).<br>**Result:** The mouse pointer changes its symbol to  .<br>**3.** Click the left mouse button and keep it pressed.<br>**Result:** The mouse pointer changes its symbol to  .<br>**4.** Drag the object to the new position and release the mouse button. | **1.** Position *(see page 870)* the cursor on the object to be moved.<br>**2.** Press the **spacebar**.<br>**3.** Press the **Shift** key and keep it pressed.<br>**4.** Move the object to the target position using the arrow keys. |
| **Result:**<br>● The selected object is moved from its original position to the target position.<br>● Graphical links can only be moved if they and their partner objects are explicitly selected.<br>● If the objects to be moved are placed on a graphical link, the link is undone and the objects are inserted into the structure.<br>● Graphical links can only be kept if they and their partner objects are explicitly selected.<br>**Note:** The move operation is also possible across sections between different opened SFC sections.<br>**Note:** When objects are moved beyond the visible window of the editor window, automatic scrolling of the window will not begin until you reach the edge of the window with the mouse pointer, not as soon as the objects are moved beyond the edge. | |

## Copying objects by moving (Drag & Drop)

Copying objects via drag & drop:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 870)* the object to be copied.<br>**2.** Position the mouse pointer on the selected object.<br><br>**Result:** The mouse pointer changes its symbol to  .<br>**3.** Click the left mouse button and keep it pressed.<br>**4.** Press the **Ctrl** key and keep it pressed.<br><br>**Result:** The mouse pointer changes its symbol to  .<br>**5.** Drag the object to the target position.<br>**6.** Release the mouse button.<br><br>**Result:**<br>● The copy of the selected object and its associated information is inserted into the target position.<br>● Graphical links can only be copied if they and their partner objects are explicitly selected.<br>● If the objects to be copied are placed on a graphical link, the link is undone and the objects are inserted into the structure.<br>● When copying steps and macro steps they are automatically assigned a unique name.<br>● If an action section is assigned to a step, only the reference to the action section is inserted into the section together with the step and not the action section itself.<br>If an action section is not declared in the current section, the action with the action section is deleted when it is inserted.<br>● If a transition section is assigned to a transition, only the reference to the transition section is inserted into the section together with the transition and not the transition section itself.<br>A transition section may only be referenced once in a project. That means if a reference is already available for a transition section during insertion, only the transition object is inserted (without a reference).<br>● When copying macro steps, only the macro step is copied and not its macro step section. A new macro step section is created instead. The names of the steps within this macro step section are automatically matched to the (new) macro step names. | - |

# Inserting Rows and Columns

## Rules for Inserting Rows

The following rules apply when inserting rows:

- The total number of rows in an SFC section cannot be changed.
  However, the objects of an SFC section can be moved downwards by rows within the available section and therefore empty rows can be inserted into the string.
- The same number of rows is inserted as selected in the vertical ruler.
- As many rows can be inserted as there are empty rows at the end of the section.
- The new rows are inserted above the first selected row (i.e. all objects of the selected rows are moved down).
- When rows are inserted within existing links, these links are kept and are adjusted to the new position of the objects.

## Inserting Rows

| Step | Action |
|---|---|
| 1 | Select *(see page 871)* the desired number of rows in the vertical ruler (click the row numbers). |
| 2 | Use the **Insert cells** menu command from the shortcut menu of the vertical ruler or the **Edit** → **Row operation** → **Insert cells** command.<br>**Result:** The number of selected rows is inserted above the first selected row. |

## Rules for Inserting Columns

The following rules apply when inserting columns:

- The total number of columns in an SFC section cannot be changed.
  However, the objects of an SFC section can be moved right by columns within the available section and therefore empty columns can be inserted into the string.
- The same number of columns is inserted as is selected in the horizontal ruler.
- As many columns can be inserted as there are empty columns at the right border of the section.
- The new columns are inserted above the first selected column (i.e. all objects of the selected columns are moved right).
- When columns are inserted within existing branches or joints, the affected branches or joints are expanded by the corresponding number of columns.

## Inserting Columns

| Step | Action |
|---|---|
| 1 | Select *(see page 871)* the desired number of columns in the horizontal ruler (click the row numbers). |
| 2 | Use the **Insert cells** menu command from the shortcut menu of the horizontal ruler or the **Edit** → **Row operation** → **Insert cells** command.<br>**Result:** The number of selected columns is inserted in front of the first selected column. |

# Deleting Rows and Columns

## Rules for Deleting Rows

The following rules apply when deleting rows:

- The total number of rows in an SFC section cannot be changed.
  However, the objects of an SFC section can be deleted by rows and the objects of the following rows can be moved up.
- The same number of rows are deleted as are selected in the vertical ruler.
- The objects of the selected row(s) are deleted and the objects of the following rows are moved up by the respective number of rows.
- If rows are deleted within existing links, these links are kept and are adjusted to the new position of the objects.

## Deleting Rows

| Step | Action |
|---|---|
| 1 | Select *(see page 871)* the desired number of rows in the vertical ruler (click the row numbers). |
| 2 | You have the following options to delete the selected rows:<br>● Select the menu command **Move and delete cells** in the rows shortcut menu,<br>● execute the command **Edit → Vertical ruler → Move and delete cells**, or<br>● Press the **Del** key.<br>**Result:** The selected number of rows is deleted. |

## Rules for Deleting Columns

The following rules apply when deleting columns:

- The total number of columns in an SFC section cannot be changed.
  However, the objects of an SFC section can be deleted by columns and the objects of the following rows can be to the left.
- The same number of columns are deleted as are selected in the horizontal ruler.
- The objects of the selected column(s) are deleted and the objects of the following columns are moved to the left by the respective number of columns.
- If columns are deleted within existing branches or joints, the affected branches or joints are contracted by the respective number of columns.

## Deleting Columns

| Step | Action |
|---|---|
| 1 | Select *(see page 871)* the desired number of columns in the horizontal ruler (click the row numbers). |
| 2 | You have the following options to delete the selected columns:<br>● Select the menu command **Move and delete cells** in the rows shortcut menu or,<br>● execute the command **Edit → Vertical ruler → Move and delete cells**, or<br>● Press the **Del** key.<br>**Result:** The selected number of columns is deleted. |

# Displaying the object properties

## Displaying the object properties

You have the following options to display the object properties dialog box:

- Double-click the object.
- Select *(see page 870)* the object and use the **Edit → Properties... dialog box.**.
- Select *(see page 870)* the object and use the **Properties...command.**from the shortcut menu.
- Select *(see page 870)* the object and then press the **Alt**+**Enter** key combination.

When the properties dialog box is called without an object being selected, the section properties dialog box *(see page 470)* is opened.

If several objects are selected, the commands do not work.

## View data properties

You have the following options to display the section properties dialog box *(see page 1197)*:

- using the data editor *(see page 325)*
  **a.** Selecting one or several lines in a list editor
  **b.** Using the shortcut menu select the **Properties**command.
- via the SFC section
  **a.** Selecting one or several transitions in the SFC section.
  **b.** From the context menu select the command **Data properties** or press **Ctrl** + **Enter**.

# Initialize Search

## Introduction

The **Initialize Search** function transfers the name of the selected element (search text) to the **Element** text box in the **Cross-References** window.

## Selecting the search text

Process the following steps to select the search text:

| Step | Action |
|---|---|
| 1 | Select the element to search for (search text).<br>The following elements can be searched for:<br>● step variables<br>(To search for a step variable, select the step symbol.)<br>● transition variables<br>(To search for a transition variable, select the transition symbol.) |
| 2 | Use the **Services** → **Initialize Search** menu command.<br>or<br>Select the menu command **Initialize search** from the sections shortcut menu.<br>or<br>Press the **Ctrl**+**U** key combination.<br>**Result:** The search text is transferred to the **Element** text box in the **Cross-References** window. |

# Initialize Animation Table

## Introduction

The **Initialize Animation Table** transfers the selected variables to the Animation table *(see page 1502)*.

## Selecting the variables

Process the following steps to select variables:

| Step | Action |
|---|---|
| 1 | Select the variables to transfer to the Animation table.<br>The following items can be selected:<br>● step variables<br>(To transfer step variables, select the step symbol.)<br>● transition variables<br>(To accept a transition variable, select the transition symbol.) |
| 2 | Use the **Services** → **Initialize Animation table** menu command.<br>or<br>Select the menu command **Animation Table** from the sections context menu.<br>or<br>Press the **Ctrl**+**T** key combination.<br>**Result:** The variable(s) is/are transferred to the Animation table *(see page 1502)*. |

## Initialize Animation Table

Using **Initialize Animation Table** (**Ctrl**+**T**) you create an animation table with a default name e.g. `Table[SFC Editor - Mixer : [MAST]]`. You can rename the table.

All the variables you selected are listed in the animation table.

Selecting a new set of variables and using **Initialize Animation Table** again, all new variables are added to this animation table (if you did not rename it).

## Initialize New Animation Table

Using **Initialize New Animation Table** (**Ctrl**+**Shift**+**T**) you create a new animation table with an indexed default name e.g. `Table[SFC Editor - Mixer : [MAST]1]`. You can rename the table.

All the variables you selected are listed in the table.

But if you click anywhere in the section and do not select any variable, you create a new empty animation table.

Every time you use **Initialize New Animation Table** you create a new animation table.

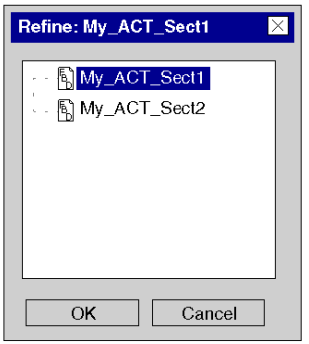
# Refining steps, macro steps and transitions

## Introduction

This function makes it possible to "look into" a macro step or a transition (section). This makes it possible to see the internal state of the macro/transition section when animation is used, or to make changes to the logic very quickly.

## Refining steps

Process the following steps to refine steps:

| Step | Action |
|---|---|
| 1 | Select the step. |
| 2 | You have the following options to display to refine steps:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** If the step contains only one action section, this will be opened automatically.<br>If the step contains several action sections, a selection dialog box is opened.<br>Example:<br>  |
| 3 | Select the desired action section.<br>**Result:** The action section is opened. |

## Refining macro steps

Process the following steps to refine macro steps:

| Step | Action |
|---|---|
| 1 | Select the macro step. |
| 2 | You have the following options to refine macro steps:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** The macro section is opened.<br>**Note:** With nested macro steps several **refine** processes are possible. |

## Refining transitions

Process the following steps to refine transitions:

| Step | Action |
|---|---|
| 1 | Select the transition. |
| 2 | You have the following options to refine transitions:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** The transition section is opened. |

# Go to

## Introduction

The **Go to** function is used to jump to:

- a specific location in the current section,
- a specific bookmark in the current section or
- a specific step or macro step in the current section or
- a specific jump in the current section.

## Open the dialog

The following options are available to open the**Go to** dialog box.

- Use the menu command **Edit** → **Go to**,
- select the menu command **Go to** from the sections shortcut menu,
- Press the key combination **Ctrl**+**G**.
  or
- Choose the  symbol.

## Jump to a Position

Go to a certain location in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 889)* the **Go to** dialog box. |
| 2 | Select the **Position** tab. |
| 3 | Enter the line and/or column number. |
| 4 | Confirm the entry using the command button **Go to**.<br>**Result:** The desired position is displayed. |

## Jump to Bookmark

Go to a certain bookmark in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 889)* the **Go to** dialog box. |
| 2 | Choose the**Bookmark** tab. |
| 3 | Select a bookmark in the list. |
| 4 | Double click on the selected bookmark or press the **Go to** command button.<br>**Result:** The bookmark is shown and marked as the current bookmark *(see page 576)*. |

### Jump to a Step

Jump to a certain step or macro step in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 889)* the **Go to** dialog box. |
| 2 | Select the **Step** tab. |
| 3 | Select a step or a macro step in the list. |
| 4 | Double click on the selected step or macro step or press the **Go to** command button.<br>**Result:** The desired step is displayed. |

### Jump to a Jump

Go to a certain jump in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 889)* the **Go to** dialog box. |
| 2 | Select the **Jump** tab. |
| 3 | Select a jump in the list. |
| 4 | Double click on the selected jump or press the **Go to** command button.<br>**Result:** The desired jump is displayed. |

# Go to dialog

## Call the dialog box:

Refer to the Opening the dialog *(see page 889)*

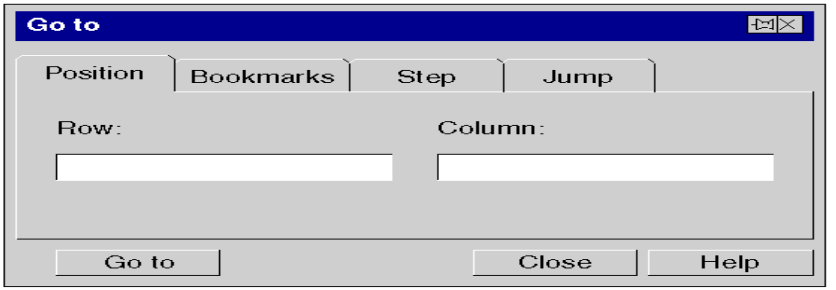
## Call the dialog box

The **Go to** dialog box is made up of three tabs:

- **Location**
  Go to a certain position in the current section using this tab:
- **Bookmarks**
  Go to a certain bookmark in the current section using this tab:
- **Step**
  This tab is used to a jump to a certain step or macro step in the current section
- **Jump**
  Go to a certain jump in the current section using this tab:

## Position tab

Representation of the tab **Position**:

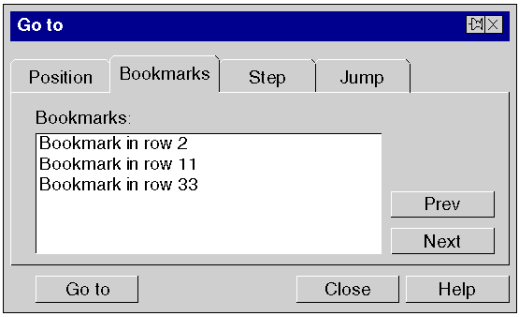



Elements of the **Position** tab

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Row** | The number of the row to be displayed can be entered in this text box. |
| **Column** | The number of the column to be displayed can be entered in this text box. |
| **Go to** | Display the current position using the button desired position *(see page 889)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

## Register tab Bookmarks

Representation of the**Bookmark** tab:

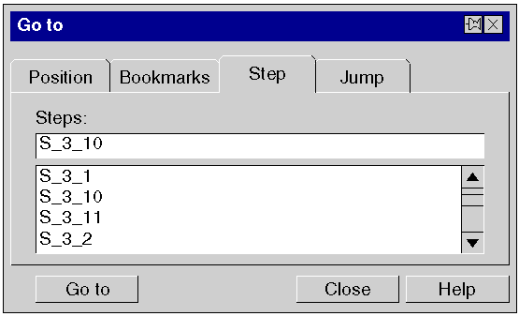



Elements of the tab **Bookmark**:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Bookmark:** | Select the bookmark to be displayed from this list box. |
| **Prev** | If a current bookmark *(see page 576)* is selected, a jump is made to the previous one and the last one is marked as the current bookmark.<br>If no current bookmark is selected, the margin is search from the bottom to the top for the next bookmark, which is marked as the current bookmark.<br>When the start of the section is reached, the search starts again from the bottom. |
| **Next** | If a current bookmark *(see page 576)* is selected, a jump is made to the previous one and the last one is marked as the next bookmark.<br>If no current bookmark is selected, the margin is search from top to bottom for the next bookmark, which is marked as the current bookmark.<br>When the end of the section is reached, the search starts again from the top. |
| **Go to** | Display the current position using the button desired bookmark *(see page 889)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

## tab Step

Display of the Register **Step**:

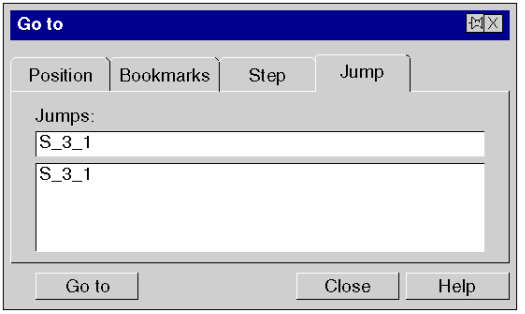



Elements of the register tab **Step**:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Steps:** | You can enter the names of the steps and macro steps to be shown in this text box.<br>You can enter the name of the step or macro step in the text box with the keyboard or using the drop-down list by clicking with the mouse. |
| **Go to** | Display the desired step or macro step *(see page 890)* using the button. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

## Tab Jump

Display of the Register **Jump**:



Elements in the **Jump**tab:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Locations:** | You can enter the names of the jump to be shown in this text box.<br>You can enter the name of the jump in the text box with the keyboard or using the drop-down list by clicking with the mouse. |
| **Go to** | Display the current position using the button desired jump *(see page 890)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

# List and display jumps

## Introduction

The **Display jumps** function is used to list all jumps referencing this step and display a selected position of the current section.

## Open the dialog

To open the dialog, select the jump and execute the **Display jumps** command in the context menu of the jump.

## List in the dialog

Since the jumps do not have their own names, the dialog lists the positions of the jumps and names of their preceding steps.

## Display a selected position

Follow the steps below to display a selected position in the current section.

| Step | Action |
|---|---|
| 1 | Open the **Display jumps** dialog box: |
| 2 | Select a position from the list. |
| 3 | Confirm the selection using the command button **Go to**.<br>**Result:** The desired position is displayed.<br>**Note:** Instead of selecting the position and clicking on the button, you can also double-click on the position. |

# Go to Step

## Introduction

The **Go to Step**function displays the referenced step of a jump.

## Display the referenced step

Perform the steps below to display the referenced step of a jump.

| Step | Action |
|---|---|
| 1 | Select a jump, |
| 2 | Open the context menu (right-click). |
| 3 | Click on **Go to Step**.<br>**Result:** The referenced step is displayed. |

# Section 23.2
## General Information about Steps

### Overview

This section provides general information about using steps in the SFC programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selecting steps | 898 |
| Placing steps | 900 |

# Selecting steps

## Introduction

An SFC section is a "status machine", i.e. the status is indicated by the active steps. Zero or more actions belong to every step.

## Selecting steps

Selecting steps:

| Step type | Description | Call via menu command | Call via symbol | Call via keyboard | Cursor symbol |
|---|---|---|---|---|---|
| "Normal" step *(see page 902)* | A step becomes active, when the upstream transition is satisfied, and is normally inactive when the downstream is satisfied.<br>Zero or more actions belong to every step. Steps without action are known as waiting steps.<br>See also section Step *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. | **Edit** → **New** → **Step**<br>or<br>**Step** menu command from the shortcut menu |  | **F3** |  |
| Initial Step *(see page 903)* | The initial status of a sequence is characterized by the initial step, which is active when initializing the project containing the SFC section. Initial steps are not usually assigned an action. With single token (conforming to IEC 61131-3), only one initial step is permitted per sequence.<br>With multi token, any number (0 up to all steps in the string) of initial steps are possible.<br>See also section Step *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. | **Edit** → **New** → **Step**<br>or<br>**Step** menu command from the shortcut menu |  | **F3** |  |

| Step type | Description | Call via menu command | Call via symbol | Call via keyboard | Cursor symbol |
|---|---|---|---|---|---|
| Macro Step *(see page 916)* | Macro steps are used to call macro sections and therefore the hierarchical structure of sequence control.<br>Macro sections calls are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow macro sections** check box.<br>See also section Macro steps and macro sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. | **Edit** → **New** → **Macro Step**<br>or<br>**Macro Step** menu command from the shortcut menu |  | **Ctrl+F3** |  |
| InStep *(see page 927)* | Every macro section begins with an InStep.<br>InSteps are created automatically in macro sections by the SFC editor and cannot be deleted, copied or inserted manually.<br>See also section InStep *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. | - | - | - | - |
| OutStep *(see page 934)* | Every macro section ends with an OutStep.<br>OutSteps are created automatically in macro sections by the SFC editor and cannot be deleted, copied or inserted manually.<br>See also section OutStep *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. | - | - | - | - |

# Placing steps

## Introduction

Steps can be placed in any free cell.

If a step is placed in a cell that is already occupied by an object, an error message is returned.

If a step is placed in a cell where the neighboring cell above or below already contains a step, an error message is returned because a transition must be placed between two steps (see also section Connection rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual).

A step which is placed always connects automatically with the neighboring objects above and below if there are no free cells between them.

## Note

In the **Tools** → **Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** check box is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the check box is **cleared**.

## Placing steps:

Placing steps:

| Step | Action |
|---|---|
| 1 | Select the desired step, see also *Selecting steps, page 898*. |
| 2 | Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The selected step is inserted. |
| 3 | To place further steps of the same type:<br>● Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Section 23.3
## Editing "normal" steps

### Overview

This section describes how to edit "normal" steps in the SFC programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Defining the properties of steps | 902 |
| Steps properties dialog box | 909 |

# Defining the properties of steps

## Introduction

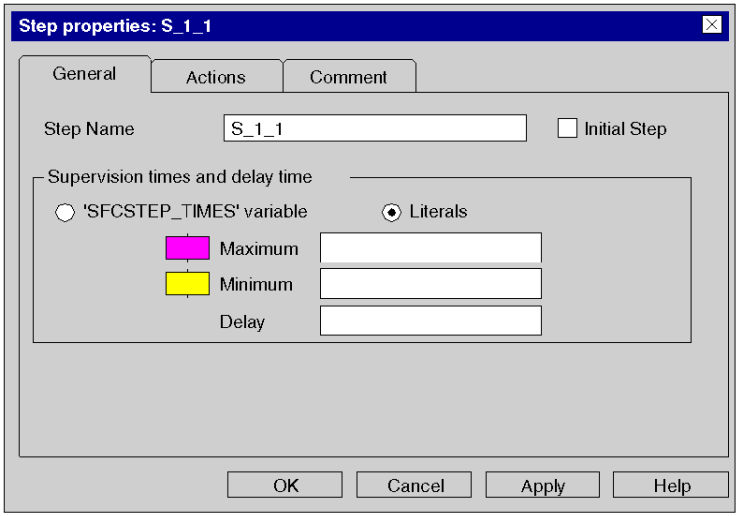
The following can be defined for steps:

- the step name *(see page 902)*
- whether the step is an initial step *(see page 903)*
- the supervision times and the delay time
  - as a literal *(see page 904)*
  - as a SFCSTEP_TIMES variable *(see page 905)*
- the actions assigned *(see page 940)*
- a comment *(see page 908)*

To find a specific jump in the current section use the **Go to** dialog box.

## Defining the step name

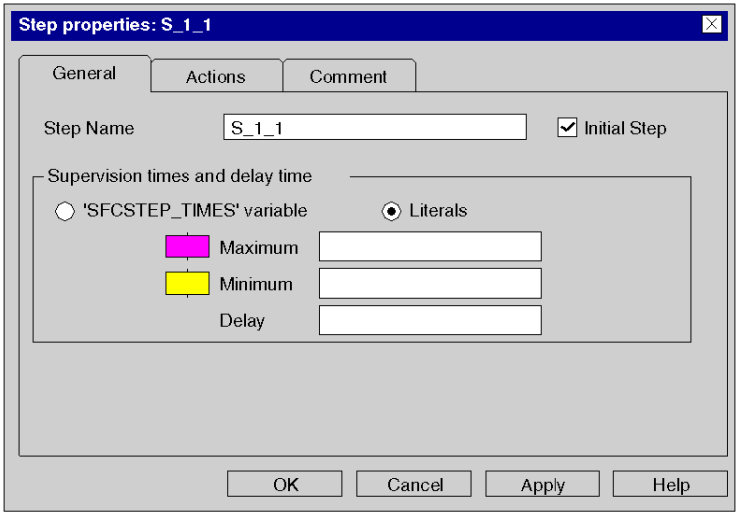
Defining the step name:

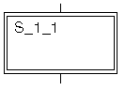
| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the step.<br>**Result**: The step properties dialog box *(see page 909)* is opened.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | In order to change the name of the step, enter a step name in the **Step Name** text box, see also section Step names *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 3 | Confirm the step names with **OK**.<br>**Result:** The selected step name is enabled, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed. |

## Defining an initial step

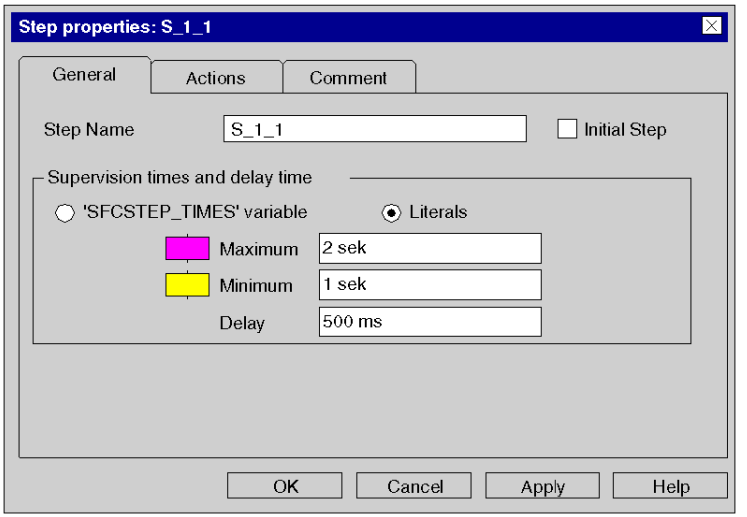
Defining an initial step:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the step.<br>**Result**: The step properties dialog box *(see page 909)* is opened. |
| 2 | In order to define an initial step, activate the **Initial Step** check box, see also section Step types *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Confirm the entry with **OK**.<br>**Result:** The selected step is enabled in the section as an initial step, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed.<br>Example:<br>  |

## Defining the supervision time/delay time as a literal

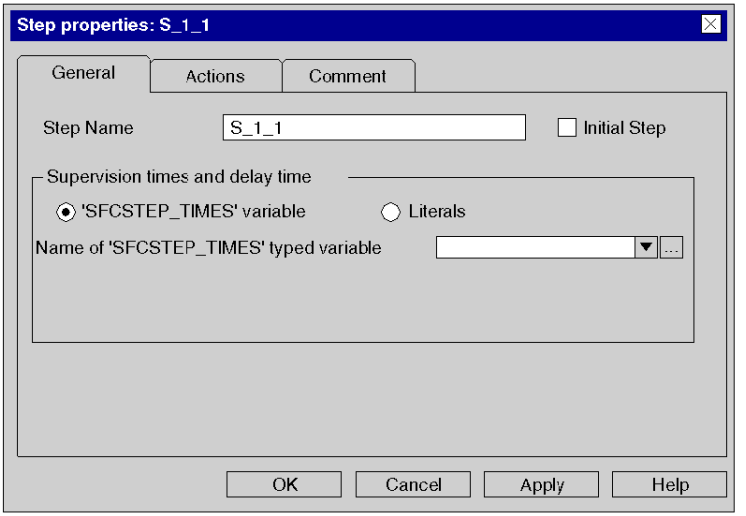
Defining the supervision time/delay time as a literal:

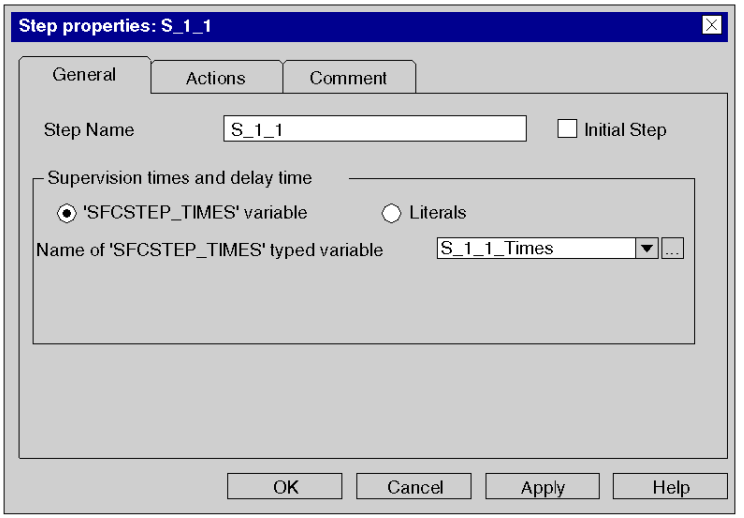
| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the step.<br>**Result**: The step properties dialog box *(see page 909)* is opened. |
| 2 | In the **Supervision times and delay time** area, activate the **Literal** option button if it isn't already activated. |
| 3 | Enter a value for the minimum supervision time, the maximum supervision time and the delay time, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>Example:<br>  |

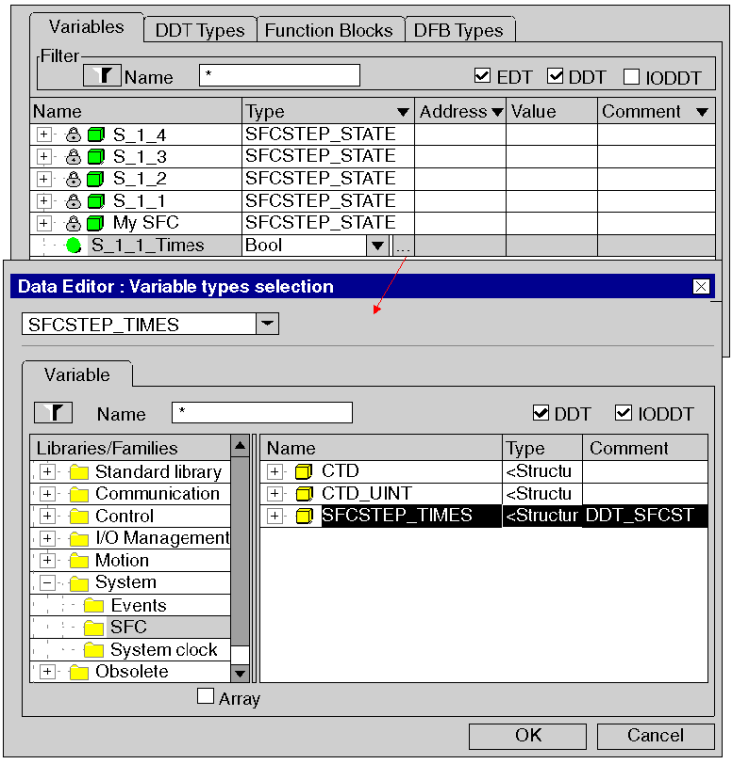
| Step | Action |
|---|---|
| 4 | Confirm the entries with **OK**.<br>**Result:** The step times are enabled, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed. |

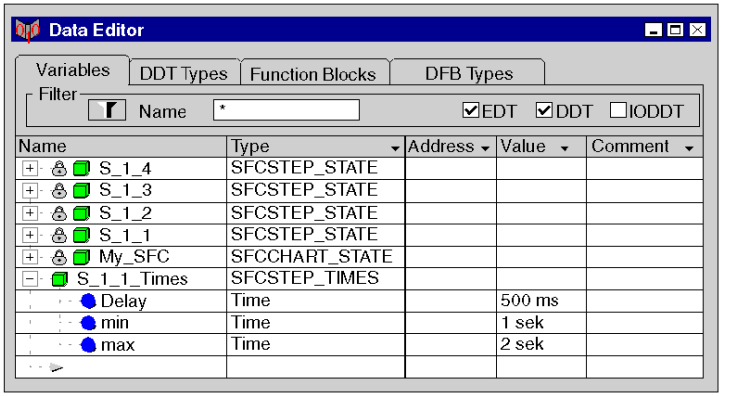
## Defining the supervision time/delay time as an SFCSTEP_TIMES variable

Defining the supervision time/delay time as an SFCSTEP_TIMES variable:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the step.<br>**Result**: The step properties dialog box *(see page 909)* is opened. |
| 2 | In the **Supervision times and delay time** area, activate the **'SFCSTEP_TIMES' variable** option button if it isn't already activated.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | In the data selection box *(see page 1911)*, specify a variable of the `SFCSTEP_TIMES` *(see Unity Pro, Program Languages and Structure, Reference Manual )* data type.<br>You have the following options to enter variables:<br>● You can enter the name of the variable.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Using the command button **...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example:<br>  |
| 4 | Confirm the entries with **OK**.<br>**Result:** The selected `SFCSTEP_TIMES` variable is enabled, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed.<br>If the variable has already been declared (see also Derived variable declaration *(see page 375)*), the procedure ends here.<br>If the `SFCSTEP_TIMES` variable has not yet been declared, process the following steps. |

| Step | Action |
|---|---|
| 5 | Declare a derived variable of the `SFCSTEP_TIMES` data type, see also Derived variable declaration *(see page 375)*.<br>Example:<br>  |

| Step | Action |
|---|---|
| 6 | Enter a value for the minimum supervision time, the maximum supervision time and the delay time, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>Example:<br>  <br>**Result:** The `SFCSTEP_TIMES` variable is declared and the step frame is displayed in black. |



## Enter a comment

Enter a comment:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the step. |
| 2 | Select the **Comment** tab and enter a step comment. |
| 3 | Confirm the comment with **OK**.<br>**Result:** The comment is accepted and the properties dialog box is closed. |

# Steps properties dialog box

## Calling the properties dialog box

see Opening the properties dialog box *(see page 884)*

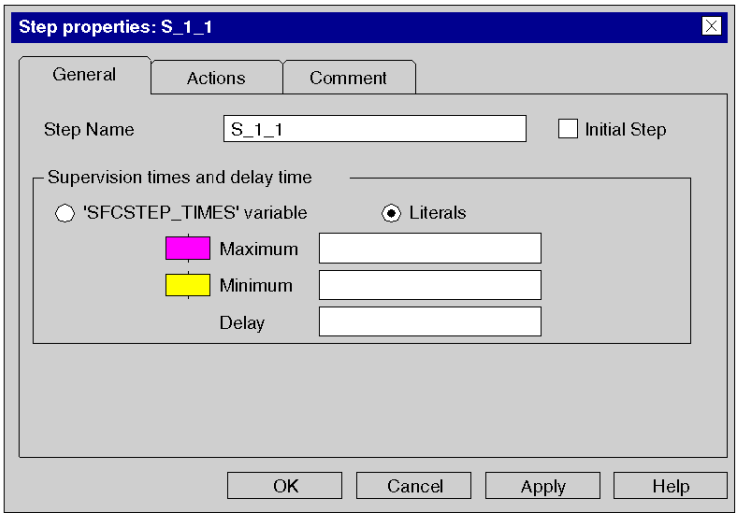
## Structure of the properties dialog box

The steps properties dialog box consists of three tabs:

- **General**
  In this tab you can define:
  - the step name *(see page 902)*
  - whether it is an initial step *(see page 903)*
  - the step supervision times and delay time as a variable *(see page 905)* or literal *(see page 904)*
- **Actions**
  In this tab you can define:
  - the actions assigned to the step as an action variable *(see page 943)* or as an action section *(see page 944)*
  - the qualifier *(see page 946)* allocated to the action
  - the time period for the qualifier *(see page 946)*
  - the execution order *(see page 947)* of the actions
- **Comment**
  In this tab, you can enter a comment *(see page 908)* about the step.

## General tab

Representation of the **General** tab:

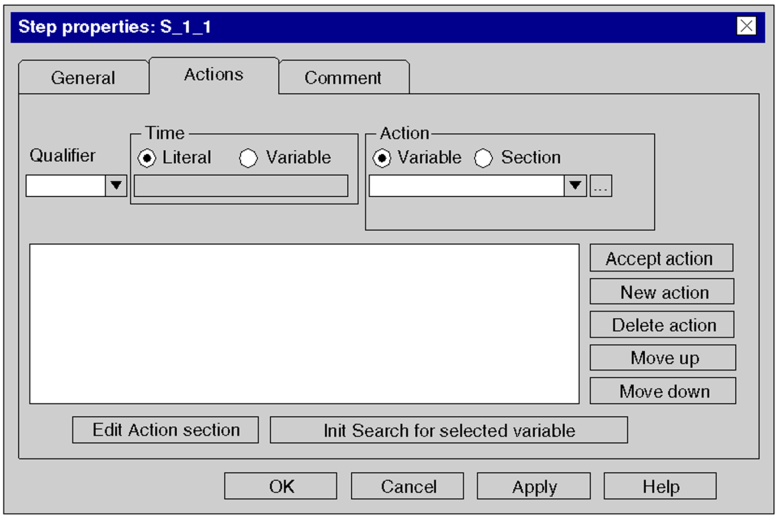


Elements of the **General** tab:

| Element | Description |
|---|---|
| **Step Name** | Enter the step name *(see page 902)* in this text box. |
| **Initial Step** | When this check box is activated, the current step becomes an initial step *(see page 903)*. |
| **Supervision times and delay time** | Define the step supervision times and delay time in this area, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>The contents of this area depends on the selected option:<br>● **'SFCSTEP_TIMES' variable**<br>When you activate this option button, you can enter the step supervision and delay times as variables *(see page 905)* of the `SFCSTEP_TIME` data type in the **Name of 'SFCSTEP_TIMES' typed variable** data selection box *(see page 1911)*, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>● **Literals**<br>When you activate this option button, you can enter the step supervision and delay times as literals *(see page 904)*, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>❍ **Maximum**<br>Enter the maximum supervision time for the current step in this text box.<br>❍ **Minimum**<br>Enter the minimum supervision time for the current step in this text box.<br>❍ **Delay**<br>Enter the delay time for the current step in this text box. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Actions tab

Representation of the register tab **Actions**:



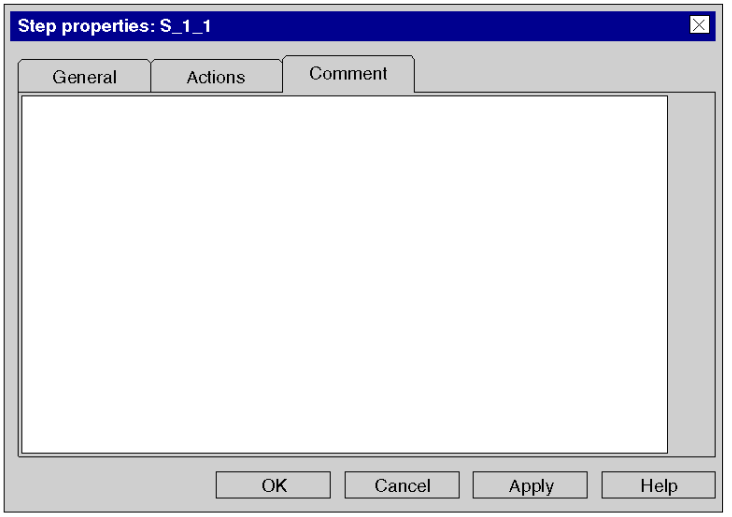

Elements of the **Actions** tab:

| Element | Description |
|---|---|
| **Qualifier** | Select the qualifier *(see page 946)* for the current action from this list box, see also section Qualifiers *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **Time** | Define the time period for the qualifiers *(see page 946)* `L` (time limited), `D` (time delayed) or `DS` (delayed and saved) in this area.<br>The contents of this area depends on the selected option:<br>● **Literal**<br>When you activate this option button, you can enter the time period as literals in the text box.<br>● **Variable**<br>When you activate this check box, you can enter the time period as a `TIME` data type in the data selection box *(see page 1911)*. |
| **Action** | Define the action in this area.<br>The contents of this area depends on the selected option:<br>● **Variable**<br>When you activate this check box, you can enter the action as an action variable *(see page 943)* in the data selection box *(see page 1911)*.<br>● **Section**<br>When you activate this check box, you can enter the action as an action section *(see page 944)* in the list box. |

| Element | Description |
|---|---|
| Action list box | All actions of the current step are displayed in this list box. |
| **Edit Action section** | This command button is only available when an action section is selected in the action list box.<br>Use this command button to close the properties dialog box and open the selected action section. |
| **Init Search for selected variable** | This command button is only available when an action variable is selected in the action list box.<br>Use this command button to close the properties dialog box and open the selected action section.<br>The name of the action variable is applied using this command button in the **Cross-references** window. |
| **Accept action** | Use this command button to replace the definitions of the selected action in the action list box with the modified definitions *(see page 942)*. |
| **New action** | Use this command button to insert the new action *(see page 940)* into the action list box. |
| **Delete action** | Use this command button to delete the selected action in the action list box. *(see page 948)*<br>**Note:** When deleting actions with action sections you are asked to confirm whether the action section should be deleted. Please note that deleted sections cannot be restored using **Edit** → **Undo**. |
| **Move up** | Use this command button to move the selected actions in the action list box one position up, see also Changing the execution order of actions *(see page 947)*. |
| **Move down** | Use this command button to move the selected actions in the action list box one position down, see also Changing the execution order of actions *(see page 947)*. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the step. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 23.4
# Editing macro steps

## Overview

This section describes how to edit macro steps in the SFC programming language.

Macro steps are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow macro sections** check box.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Defining the properties of macro steps | 916 |
| Macro Steps Properties Dialog Box | 923 |

# Defining the properties of macro steps

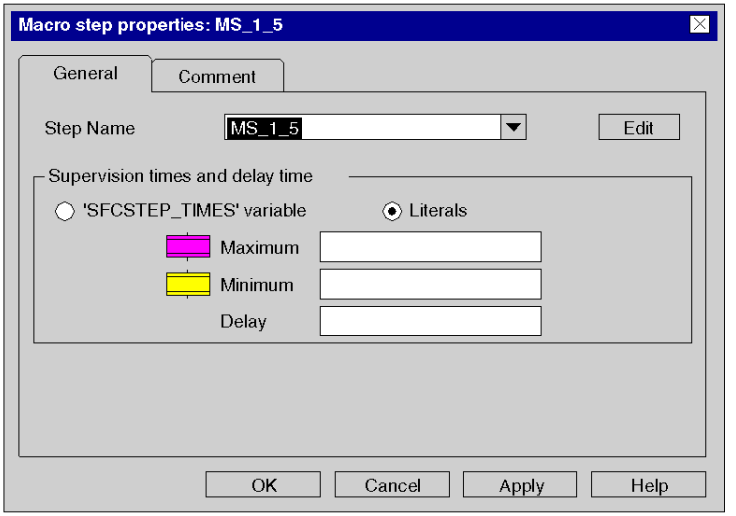
## Introduction

The following can be defined for macro steps:

- the macro step name *(see page 916)*
- the logic of the macro section *(see page 917)*
- the supervision times and the delay time
  - as a literal *(see page 918)*
  - as a SFCSTEP_TIMES variable *(see page 919)*
- a comment *(see page 922)*

## Defining the macro step name

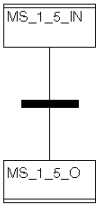
Defining the macro step name:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the macro step.<br>**Result:** The macro step properties dialog box is opened, see also *Macro Steps Properties Dialog Box, page 923*.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | In order to change the name of the step, enter a name in the **Step Name** text box, see also section Macro step names *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 3 | Confirm the macro step name with **OK**.<br>**Result:** The macro step name selected is enabled for the macro step and the steps within the macro step section. The properties dialog box is closed and a syntax and semantics check *(see page 867)* is performed. |

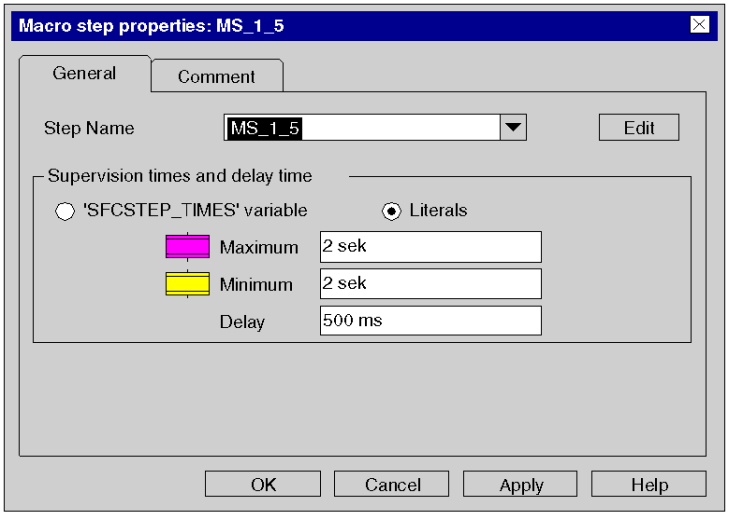
## Editing the macro section

Editing the macro section:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* of the macro step, and press the **Edit** command button.<br>or<br>Select the macro step and process the **Services** → **Refine** menu command.<br>**Result:** The macro step properties dialog box is closed and the macro section of the macro step is opened.<br>Example:<br>  |
| 2 | Create the macro section logic, see also section Macro sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |

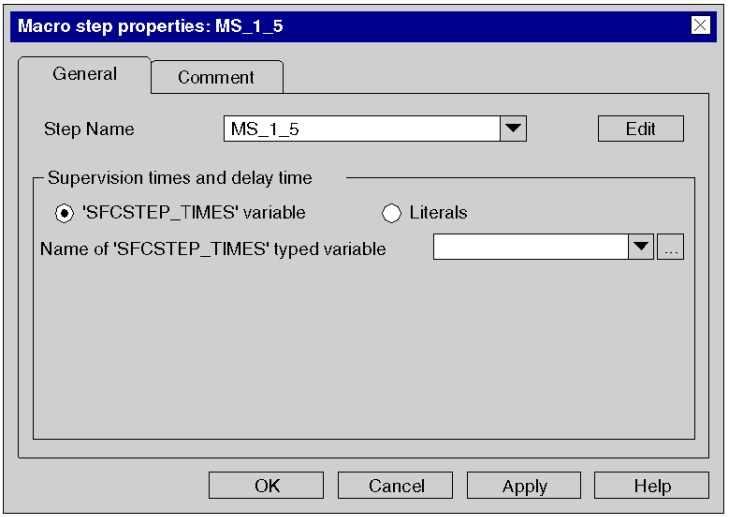
## Defining the supervision time/delay time as a literal

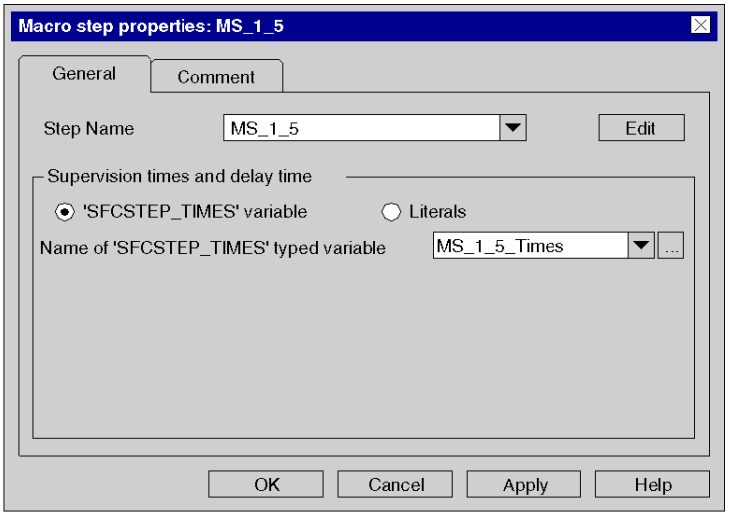
Defining the supervision time/delay time as a literal:

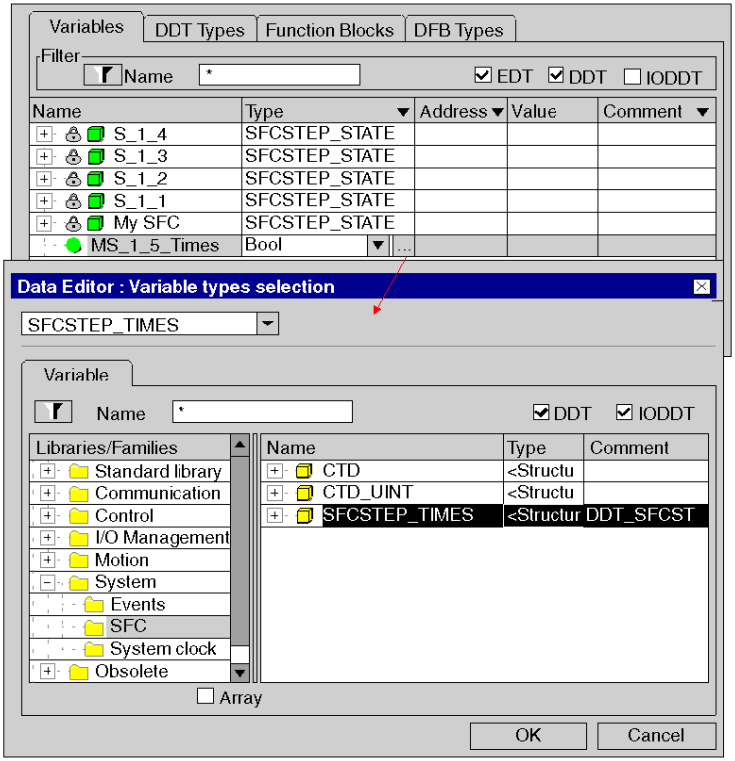
| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the macro step.<br>**Result:** The macro step properties dialog box is opened, see also *Macro Steps Properties Dialog Box, page 923*. |
| 2 | In the **Supervision times and delay time** area, activate the **Literal** option button if it isn't already activated. |
| 3 | Enter a value for the minimum supervision time, the maximum supervision time and the delay time, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>Example:<br>  |
| 4 | Confirm the entries with **OK**.<br>**Result:** The step times are enabled, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed. |

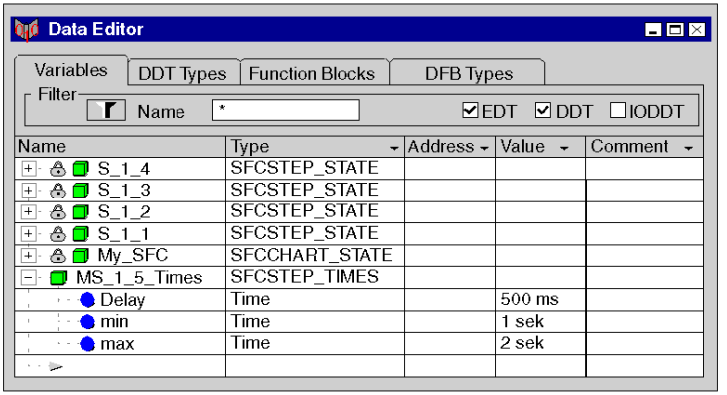
## Defining the supervision/delay time as an SFCSTEP_TIMES variable.

Defining the supervision time/delay time as an SFCSTEP_TIMES variable:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the macro step.<br>**Result:** The macro step properties dialog box is opened, see also *Macro Steps Properties Dialog Box, page 923*. |
| 2 | In the **Supervision times and delay time** area, activate the **'SFCSTEP_TIMES' variable** option button if it isn't already activated.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | In the data selection box *(see page 1911)*, specify a variable of the SFCSTEP_TIMES data type, see also section SFCSTEP_TIMES *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>You have the following options to enter variables:<br>● You can enter the name of the variable.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Using the command button **...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example:<br>  |
| 4 | Confirm the entries with **OK**.<br>**Result:** The selected SFCSTEP_TIMES variable is enabled, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed.<br>If the variable has already been declared (see also Derived variable declaration *(see page 375)*), the procedure ends here.<br>If the SFCSTEP_TIMES variable has not yet been declared, process the following steps. |

| Step | Action |
|---|---|
| 5 | Declare a derived variable of the `SFCSTEP_TIMES` data type, see also Derived variable declaration *(see page 375)*.<br>Example:<br>  |

| Step | Action |
|---|---|
| 6 | Enter a value for the minimum supervision time, the maximum supervision time and the delay time, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>Example:<br>  <br>**Result:** The `SFCSTEP_TIMES` variable is declared and the macro step frame is displayed in black. |



## Enter a comment

Enter a comment:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the macro step. |
| 2 | Select the **Comment** tab and enter a macro step comment. |
| 3 | Confirm the comment with **OK**.<br>**Result:** The comment is accepted and the properties dialog box is closed. |

# Macro Steps Properties Dialog Box

## Calling the properties dialog box

see *Displaying the object properties, page 884*

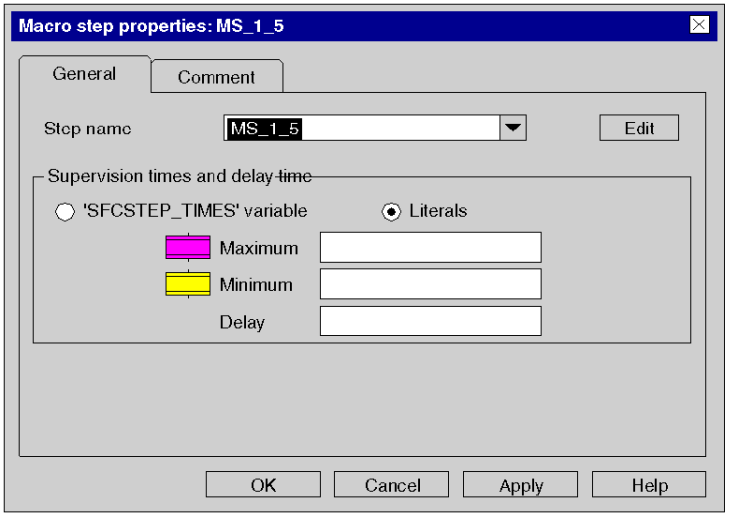
## Structure of the properties dialog box

The macro steps properties dialog box consists of 3 tabs:

- **General**
  In this tab you can define:
  - the macro step names *(see page 916)*
  - the step supervision time and delay time as a variable *(see page 919)* or literal *(see page 918)*
- **Comment**
  A comment *(see page 922)* about the step can be entered in this tab.

## Tab General

Representation of the **General** tab:

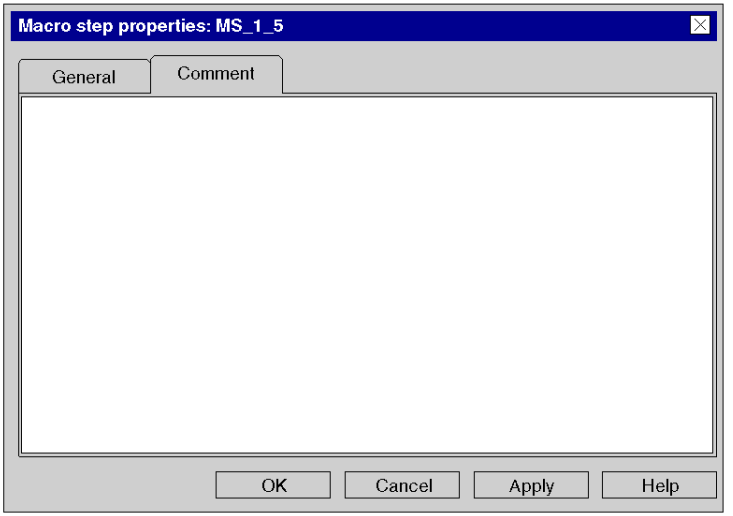


Items of the **General** tab:

| Item | Description |
|---|---|
| **Step name** | Enter the macro step name *(see page 916)* in this text box. |
| **Edit** | Use this command button to close the properties dialog box and open the macro step section. |
| **Supervision times and delay time** | You define the step supervision time and delay time in this range, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>The contents of this area depends on the option selected:<br>● **'SFCSTEP_TIMES' variable**<br>If you check this option button you can enter in the data selection box *(see page 1911)* **Name of 'SFCSTEP_TIMES' typed variables** the Step supervision and delay times as variables *(see page 919)* of the `SFCSTEP_TIME` data type.<br>● **Literals**<br>If you check this option button you can enter the Step supervision and delay times as literals *(see page 918)*.<br>❍ **Maximum**<br>Enter the maximum supervision time for the current macro step in this text box.<br>❍ **Minimum**<br>Enter the minimum supervision time for the current macro step in this text box.<br>❍ **Delay**<br>Enter the delay time for the current macro step in this text box. |
| **OK** | This command button enables all entries and closes the dialog box. |
| **Apply** | This command button enables all entries without closing the properties dialog box. |

## Tab Comment

Representation of the **Actions** tab:




Items of the **Comment** tab:

| Item | Description |
|---|---|
| Text box | Enter a comment about the step. |
| **OK** | This command button enables all entries and closes the dialog box. |
| **Apply** | This command button enables all entries without closing the properties dialog box. |

# Section 23.5
# Editing input steps

## Overview

This section describes how to edit input steps in the SFC programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Defining the properties of InSteps | 927 |
| InSteps properties dialog box | 929 |

# Defining the properties of InSteps

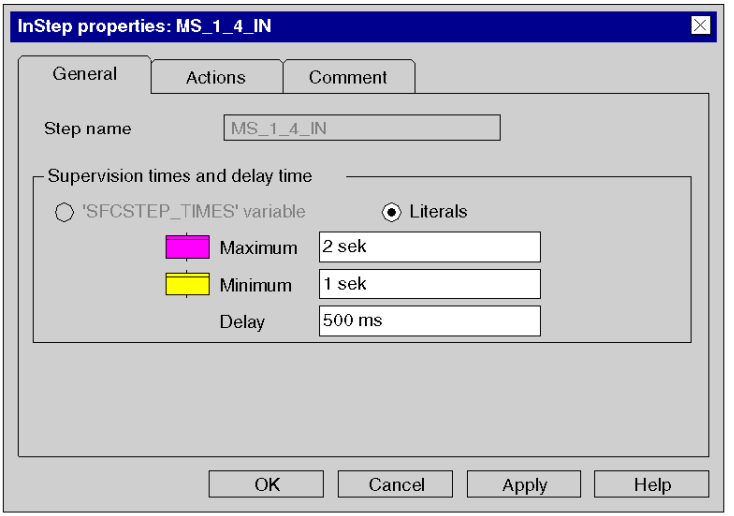
## Introduction

The following can be defined for InSteps:

- the supervision times and the delay time *(see page 927)*
- the actions assigned *(see page 940)*
- a comment *(see page 928)*

## Defining the supervision time/delay time

Defining the supervision time/delay time as a literal:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the InStep.<br>**Result**: The properties dialog box for the step is opened, see also *InSteps properties dialog box, page 929*. |
| 2 | Enter a value for the minimum supervision time, the maximum supervision time and the delay time, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>Example:<br>  |
| 3 | Confirm the entries with **OK**.<br>**Result:** The step times are enabled, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed. |

## Entering a comment

Entering a comment:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the InStep. |
| 2 | Select the **Comment** tab and enter a step comment. |
| 3 | Confirm the comment with **OK**.<br>**Result:** The comment is enabled and the properties dialog box is closed. |

# InSteps properties dialog box

## Calling the properties dialog box

see *Displaying the object properties, page 884*

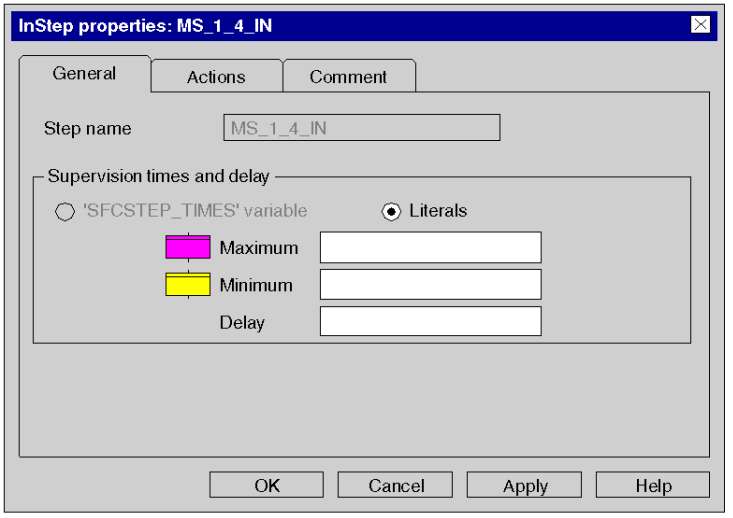
## Structure of the properties dialog box

The InStep properties dialog box consists of three tabs:

- **General**
  In this tab you can define:
  - the step supervision time and the delay time *(see page 927)*
- **Actions**
  In this tab you can define:
  - the actions assigned to the step as an action variable *(see page 943)* or as an action section *(see page 944)*
  - the qualifier *(see page 946)* allocated to the action
  - the time period for the qualifier *(see page 946)*
  - the execution order *(see page 947)* of the actions
- **Comment**
  In this tab, comment *(see page 928)* about the InStep can be entered.

## General tab

Representation of the **General** tab:

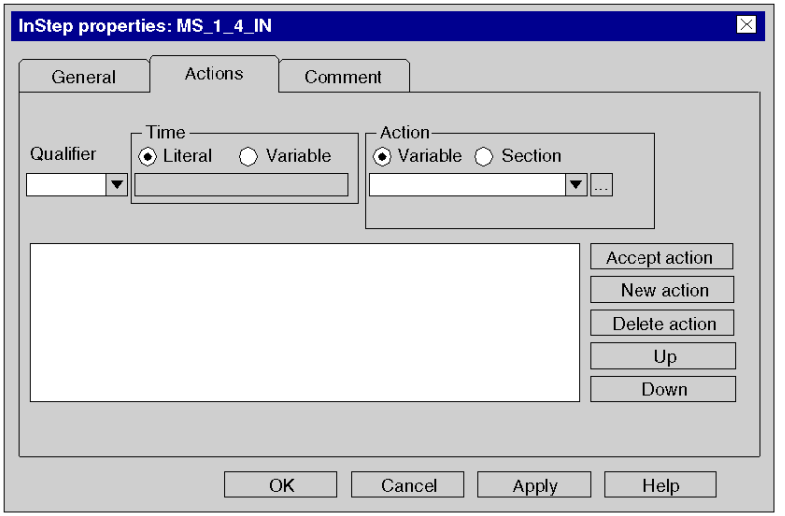


Elements of the **General** tab:

| Element | Description |
|---|---|
| **Step Name** | The names of InSteps cannot be changed, see also section Step names *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **Supervision times and delay time** | Define the step supervision times and the delay time in this area, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. For InSteps, the steps times can be exclusively defined as literals *(see page 904)*.<br>● **Maximum**<br>Enter the maximum supervision time for the current InStep in this text box.<br>● **Minimum**<br>Enter the minimum supervision time for the current InStep in this text box.<br>● **Delay**<br>Enter the delay time for the current InStep in this text box. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Actions tab

Representation of the **Actions** tab:

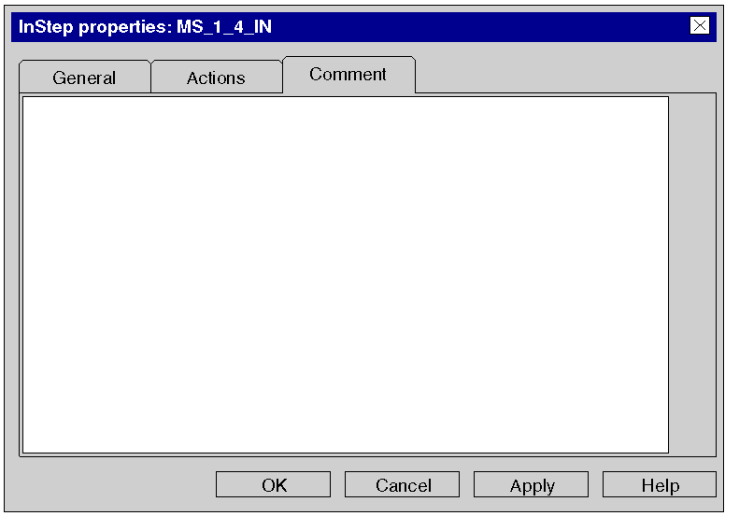


Elements of the **Actions** tab:

| Element | Description |
|---|---|
| **Qualifier** | Select the qualifier *(see page 946)* for the current action from this list box, see also section Qualifiers *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **Time** | Define the time period for the qualifiers *(see page 946)* `L` (time limited), `D` (time delayed) or `DS` (delayed and saved) in this area.<br>The contents of this area depends on the option:<br>● **Literal**<br>When you activate this option button, you can enter the time period as literals in the text box.<br>● **Variable**<br>When you activate this check box, you can enter the time period in the data selection box *(see page 1911)* as a `TIME` data type. |
| **Action** | Define the action in this area.<br>The contents of this area depends on the selected option:<br>● **Variable**<br>When you activate this check box, you can enter the action as an action variable *(see page 943)* in the data selection box *(see page 1911)*.<br>● **Section**<br>When you activate this check box, you can enter the action as an action section *(see page 944)* in the list box. |
| Action list box | All actions of the current step are displayed in this list box. |
| **Edit Action section** | This command button is only available when an action section is selected in the action list box.<br>Use this command button to close the properties dialog box and open the action section selected. |
| **Accept action** | Use this command button to replace the definitions of the selected action in the action list box with the modified definitions *(see page 942)*. |
| **New action** | Use this command button to insert the new action *(see page 940)* in the action list box. |
| **Delete action** | Use this command button to delete the selected action in the action list box. *(see page 948)*<br>**Note:** When deleting actions with action sections you are asked to confirm whether the action section should be deleted. Note that deleted sections cannot be restored using the **Edit** → **Undo** menu command. |
| **Up** | Use this command button to move the selected actions in the action list box one position up, see also *Modifying the execution order of actions, page 947*. |
| **Down** | Use this command button to move the selected actions in the action list box one position down, see also *Modifying the execution order of actions, page 947*. |
| **OK** | Us this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the InStep. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 23.6
## Editing Output Steps

### Overview

This section describes how to edit output steps in the SFC programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Defining the properties of OutSteps | 934 |
| OutStep properties dialog box | 936 |

# Defining the properties of OutSteps

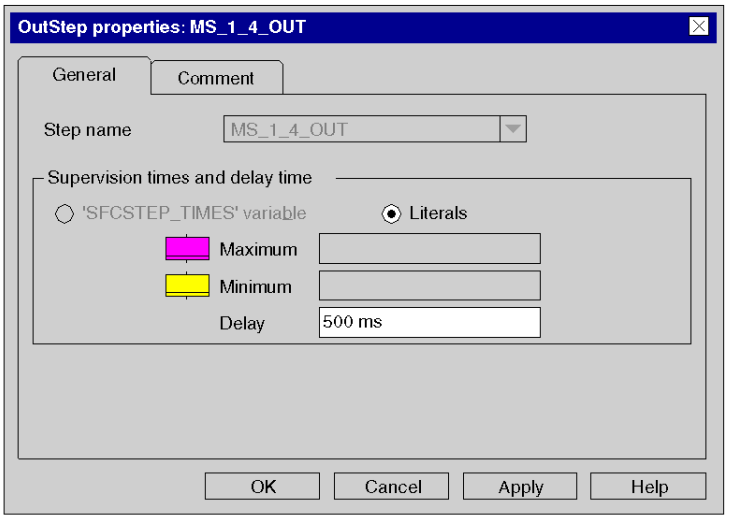
## Introduction

The following can be defined for OutSteps:

- the delay time *(see page 934)*
- a comment *(see page 935)*

## Defining the delay time

Defining the delay time as a literal:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the OutStep.<br>**Result**: The properties dialog box for the step is opened, see also *OutStep properties dialog box, page 936*. |
| 2 | Enter the value of the delay time, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>Example:<br>  |
| 3 | Confirm the entries with **OK**.<br>**Result:** The delay time is enabled, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed. |

## Entering a comment

Entering a comment:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the OutStep. |
| 2 | Select the **Comment** tab and enter a step comment. |
| 3 | Confirm the comment with **OK**.<br>**Result:** The comment is enabled and the properties dialog box is closed. |

# OutStep properties dialog box

## Calling the properties dialog box

see *Displaying the object properties, page 884*

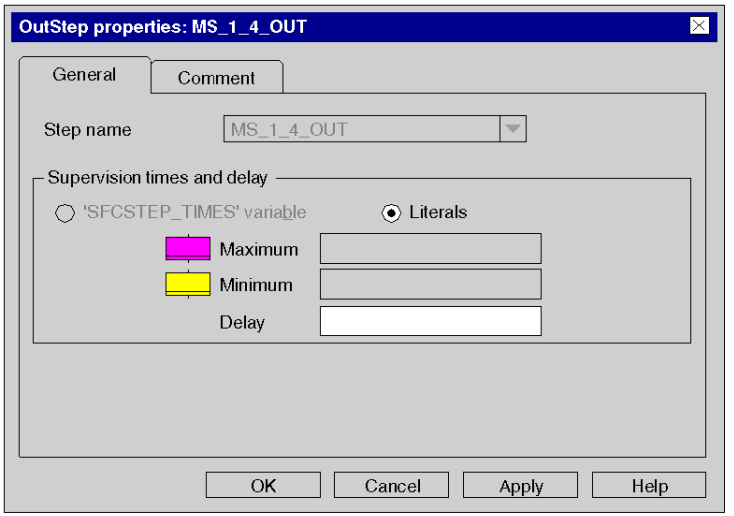
## Structure of the properties dialog box

The OutStep properties dialog box consists of two tabs:

- **General**
  In this tab you can define:
  - the delay time *(see page 927)*
- **Comment**
  In this tab, a comment *(see page 928)* about the OutStep can be entered.

## General tab

Representation of the **General** tab:

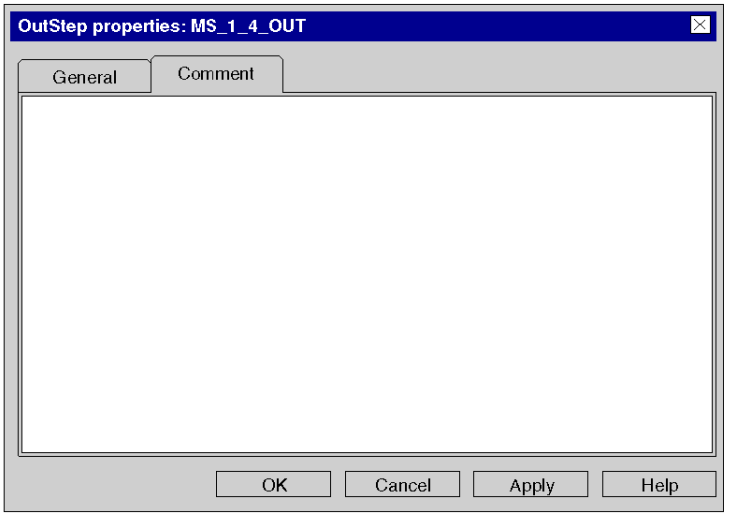


Elements of the **General** tab:

| Element | Description |
|---|---|
| **Step Name** | The names of OutSteps cannot be changed, see also section Step names *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **Supervision times and delay time** | Define the delay time in this area, see also section Step times *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. For OutSteps, the delay time can be exclusively defined as literal *(see page 934)*.<br>● **Maximum**<br>Not available<br>● **Minimum**<br>Not available<br>● **Delay**<br>Enter the delay time for the current OutStep in this text box. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:



Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the OutStep. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 23.7
## Editing actions

### Overview

This section describes how to edit actions in the SFC programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Assigning actions to a step | 940 |
| Display modes for actions | 949 |

# Assigning actions to a step

## Introduction

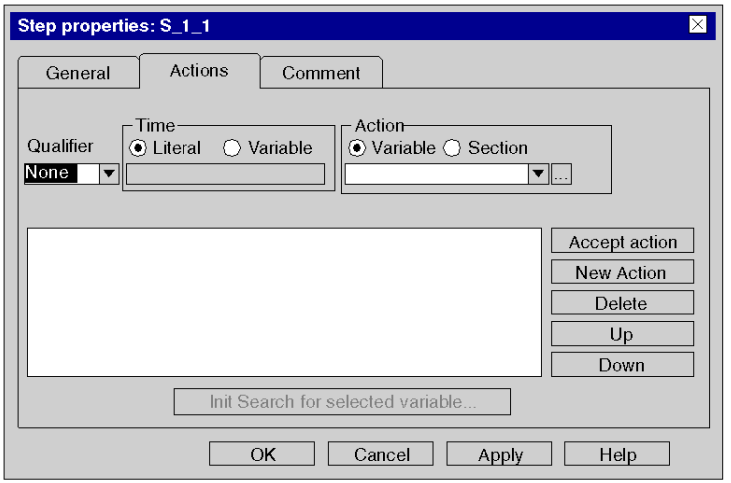
Each step contains zero or a maximum of 20 actions, see also section Actions *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

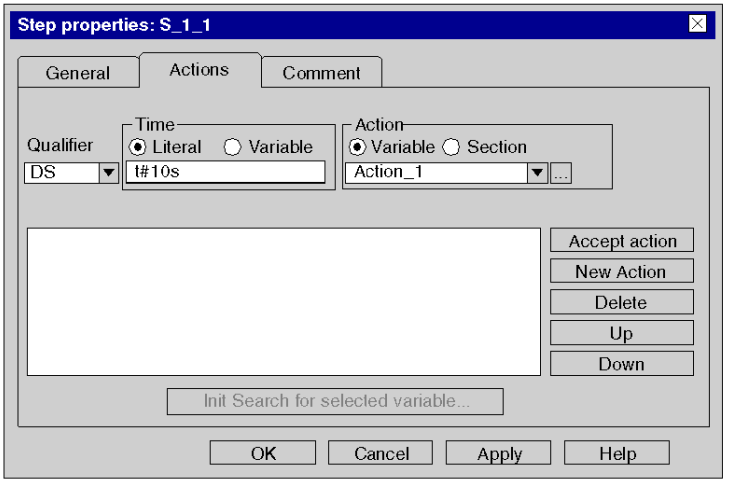
The following can be defined for actions:

- as action variable *(see page 943)*, see also section Action variables *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual
- as action section *(see page 944)* of the programming languages FBD, LD, IL or ST, see also section Action sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual
- as Qualifiers *(see page 946)*, see also section Qualifiers *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual
- the order *(see page 947)* in which the defined actions should be processed

## Defining a new action

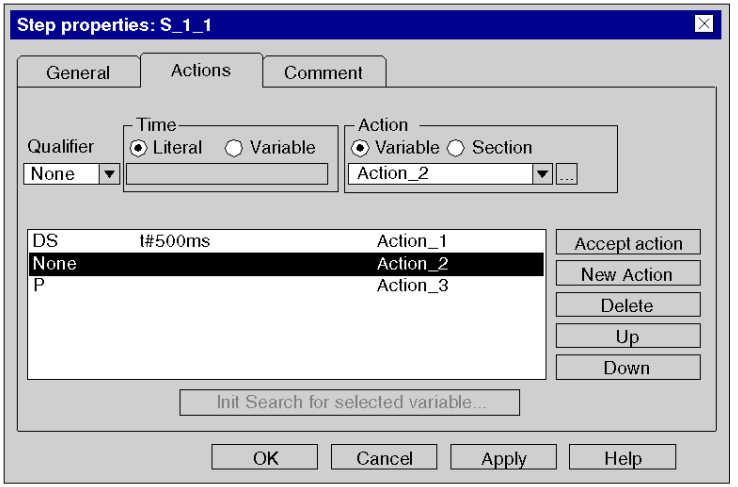
Defining a new action:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the step.<br>**Result**: The step properties dialog box is opened, see also *Steps properties dialog box, page 909*. |
| 2 | Select the **Actions** tab.<br>**Result:** The dialog box for defining actions is opened.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | In the **Action** area, define an action as<br>● action variable *(see page 943)*<br>or<br>● action section *(see page 944)* |
| 4 | If necessary, define an **Qualifier** in the Qualifier symbol *(see page 946)* list box. |
| 5 | Confirm the definitions for the action with the **New action** button.<br>**Result:** The action is enabled in the action list box.<br>Example:<br>  |
| 6 | Confirm the entries with **OK**.<br>**Result:** The actions are accepted, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed. |

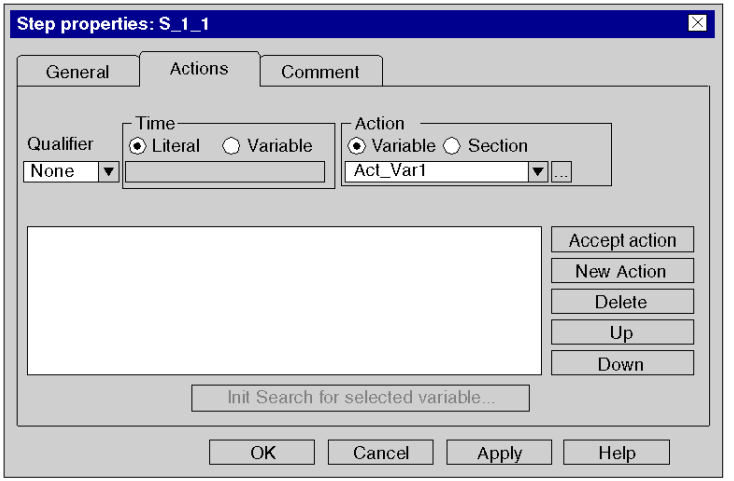
## Modifying an action

Modifying an action:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the step.<br>**Result:** The step properties dialog box is opened, see also *Steps properties dialog box, page 909*. |
| 2 | Select the **Actions** tab.<br>**Result:** The dialog box for defining actions is opened. |
| 3 | Select the action to modify from the actions list box.<br>**Result:** The properties of the selected action are enabled in the **Qualifier**, **Time** and **Action** areas.<br>Example:<br>  |
| 4 | If necessary, modify in the **Action** area<br>● action variable *(see page 943)*<br>or<br>● action section *(see page 944)*<br>**Note:** When modifying action sections you are asked to confirm whether the current action section should be deleted. Please note that deleted sections cannot be undone using**Edit** → **Undo** . |
| 5 | If necessary, define an **Qualifier** in the Qualifier symbol *(see page 946)* list box. |
| 6 | Confirm the changes for the action with the **Accept action** button.<br>**Result:** The definitions of the selected action in the action list box are replaced with the changed definitions. |
| 7 | Confirm the entries with **OK**.<br>**Result:** The actions are accepted, the properties dialog box is closed, and a syntax and semantics check *(see page 867)* is performed. |

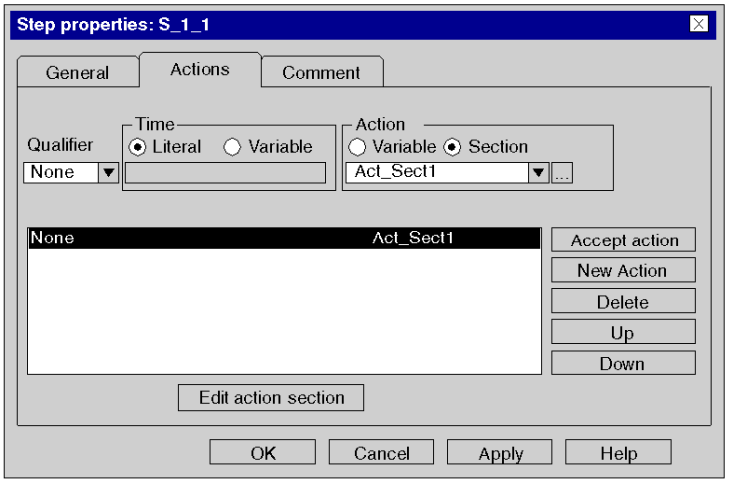
## Defining an action variable

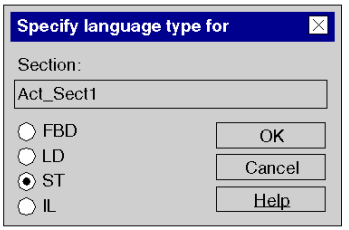
Defining an action variable:

| Step | Action |
|---|---|
| 1 | Select the **Actions** tab from the step properties dialog box, if it isn't already activated. |
| 2 | In the **Action** area, activate the **Variable** option button, if it isn't already activated. |
| 3 | Enter a Boolean variable/address, or a Boolean element of a multi-element variable in the data selection box *(see page 1911)*, see also section Action variables *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>You have the following options to enter variables:<br>● You can enter the name of the variable.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Using the **button...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example:<br>  |
| 4 | If necessary, define an qualifier *(see page 946)* for the action.<br>If you do not define an qualifier, the action section is automatically assigned the qualifier N (no qualifier). |
| 5 | Click the **New action** button to insert a new action *(see page 940)*, or the **Accept action** button to modify an existing action *(see page 942)*.<br>**Result:** The action is enabled in the action list box. |
| 6 | If the action variable has not been declared, declare it as BOOL data type, see also *Creation of EDT type variable instances, page 370*. |

## Defining a new action section

Defining a new action section:

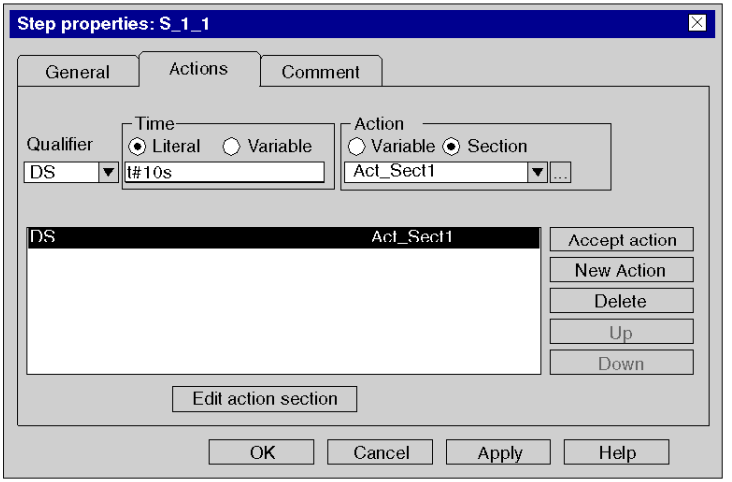
| Step | Action |
|---|---|
| 1 | Select the **Actions** tab from the step properties dialog box, if it isn't already activated. |
| 2 | In the **Action** area, activate the **Section** option button. |
| 3 | Enter the name of the action section in the list box, see also section Action sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 4 | If necessary, define an qualifier *(see page 946)* for the action section.<br>If you do not define an qualifier, the action section is automatically assigned the qualifier `N` (no qualifier). |
| 5 | Click the **New action** button to insert a new action *(see page 940)*, or the **Accept action** button to modify an existing action *(see page 942)*.<br>**Result:** The action is enabled in the action list box.<br>Example:<br>  |

| Step | Action |
|---|---|
| 6 | Click the **Edit Action section** button to create the action section logic.<br>**Result:** A dialog box is opened for defining the programming language for the action section.<br>Example:<br>  |
| 7 | Select the desired programming language for the action section and confirm the selection with **OK**.<br>**Information for creating LD sections:** An LD section contains 11-64 columns and 17-2000 lines. The number of columns is defined using the **Tools → Project Settings...dialog box, in the** Editors **tab, in the** Number of columns **text box**. These settings are only valid for new sections and not for existing sections. Always define the number of columns before you create a new LD section. (The number of lines can be increased or reduced during the creation of section contents.)<br>**Result:** The step properties dialog box is closed and the action section is opened. |
| 8 | Create the action section logic, see also section Action sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>**Note:** If the action sections subprograms are called in the **Tools → Project settings... → Language extensions** dialog box, the options **Subprograms allowed** and **Multi Token allowed** must be activated.<br>Please note that the subroutine called from the action section do **not** affect the sequence string control, i.e.<br>● the qualifier assigned to the action section has no effect on the subroutine,<br>● the subroutine remains active when the calling step is deactivated. |

## Defining the qualifier

Defining the qualifier for an action:

| Step | Action |
|---|---|
| 1 | Select the **Actions** tab from the step properties dialog box, if it isn't already activated. |
| 2 | Select the desired qualifier from the **Qualifier** list box, see also section Qualifiers *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 3 | If you select an qualifier of type `L` (time limited), `D` (time delayed) or `DS` (delayed and saved), you are assigning a time period for the qualifier.<br>You can define the time period as a literal or as a variable of the `TIME` data type:<br>● **Literal**<br>**a.** In the **Time** area, activate the **Literal** option button, if it isn't already activated.<br>**b.** Enter a time period, e.g. `500 ms`, `1 sec`.<br>● **Variable**<br>**a.** Activate the **Variable** option button in the **Time** area.<br>**b.** You have the following options to enter variables:<br>You can enter the name of the variable.<br>You can select the name of the variable from the list of recently used names using the ⊡ symbol.<br>or<br>Using the **button...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>**c.** If the variable is not already declared, declare it as a variable of the `TIME` data type and assign the variables with the desired time period, see also *Creation of EDT type variable instances, page 370*. |
| 4 | Define the action variable *(see page 943)* or the action section *(see page 944)*. |

| Step | Action |
|---|---|
| 5 | Click the **New action** button to insert a new action *(see page 940)*, or the **Accept action** button to modify an existing action *(see page 942)*.<br>**Result:** The action is enabled with the qualifier in the action list box.<br>Example:<br>  |

## Modifying the execution order of actions

Modifying the execution order of actions:

| Step | Action |
|---|---|
| 1 | Select the **Actions** tab from the step properties dialog box, if it isn't already activated. |
| 2 | Select the action from the action list box whose execution order you want to change.<br>**Note:** If several actions are assigned to a step, they are executed in the order they are listed in the action list box.<br>**Exception:** Regardless of their position in the action list box, actions with the qualifier `P1` are always processed first and actions with the qualifier `P0` are always processed last, see also section Qualifiers *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| 3 | Move the selected action to the target position using the **Move up** and **Move down** buttons. |

## Deleting actions

Deleting actions:

| Step | Action |
|---|---|
| 1 | Select the **Actions** tab from the step properties dialog box, if it isn't already activated. |
| 2 | Select the action to be deleted from the actions list box.<br>The following selection options are available:<br>● Selecting an action<br>**a.** Click the action.<br>● Selecting several non-sequential actions<br>**a.** Press the **Shift** key and keep it pressed.<br>**b.** Click the first action to be selected.<br>**c.** Click the last action to be selected.<br>● Selecting several sequential actions<br>**a.** Press the **Ctrl** key and keep it pressed.<br>**b.** Click the actions to be selected. |
| 3 | To delete the selected action(s), click the **Delete action** button.<br>**Note:** When deleting actions with action sections you are asked to confirm whether the action section should be deleted. Please note that deleted sections CANNOT be restored using **Edit** → **Undo**. |

# Display modes for actions

## Introduction

The following display modes are available:

- Input mode
- Expanded display

You have the following options to switch between modes:

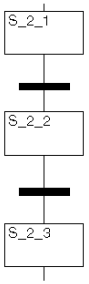
- Use the **View** → **Expanded Display** menu command.
  or
- Press the **Ctrl**+**E** key combination.

You can see which mode is active (Expanded Display) by way of a check-symbol in front of the menu command.

## Input mode

Only the step symbol is displayed in this mode.

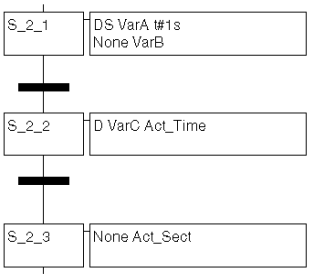
Example of a section in input mode:



## Expanded display

In addition to the step symbol, this mode displays the actions assigned to the section and their properties (qualifier, name of the action variable/section and the time period defined for the qualifier, if available). If more than two actions are assigned to a step you can scroll the list.

Example of a section in Expanded Display:



# Section 23.8
# Editing Transitions

## Overview

This section describes how to edit transitions in the SFC programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Placing transitions | 952 |
| Defining transition properties | 954 |
| Properties dialog box for transitions | 960 |

# Placing transitions

## Introduction

A transition provides the condition through which the checks of one or more pre-transition steps pass on one or more consecutive steps along the corresponding link.

Transitions can be placed in any free cell.

If a transition is placed in a cell that is already occupied by an object, an error message is returned.

If a transition is placed in a cell where the neighboring cell above or below already contains a transition, an error message is returned because a step must be placed between two transitions (see also section Connection rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual).

A transition which is placed always connects automatically with the neighboring objects above and below if there are no free cells between them.

## Activating transition placement mode

You have the following options to activate the transition placement mode:

- Use the **Edit** → **New** → **Transition** menu command.
- Use the **Transition** menu command from the shortcut menu.
- Press the **Shift**+**F3** key combination.
  or
- Select the  symbol.

The active transition placement mode is indicated by the  cursor symbol.

## Note

In the **Tools** → **Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** check box is selected, then the corresponding properties dialog box is automatically opened when an object is placed. The procedures explained here are used when the check box is **cleared**.

## Placing transitions

Placing transitions:

| Step | Action |
|---|---|
| 1 | Activate the transition placement mode. |
| 2 | Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The transition is inserted and the selection mode is active again. |
| 3 | To insert further transitions:<br>● Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Defining transition properties

## Introduction

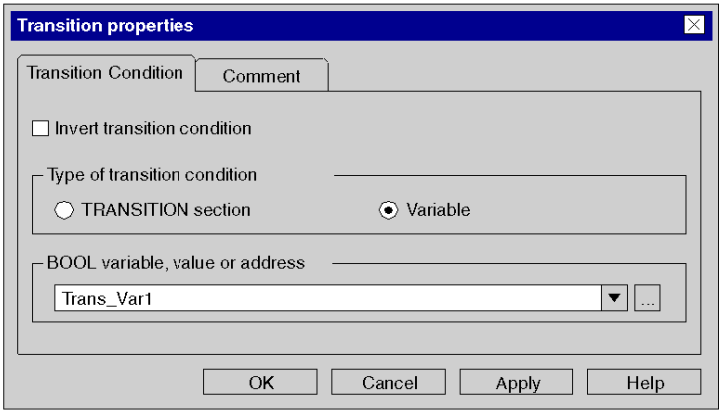
The following can be defined for transitions:

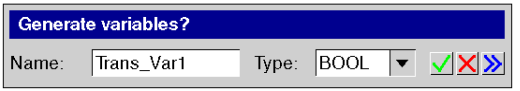
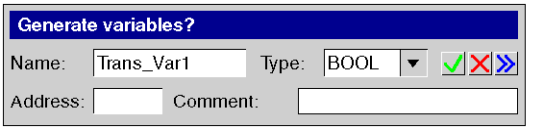
- the assigned transition condition
  - as variable *(see page 954)*
  - as transition section *(see page 957)*
- whether the transition condition should be inverted *(see page 959)* (trigger the transition when 0)
- a comment *(see page 959)*

## Defining the transition condition as a variable

Defining the transition condition as a variable:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the transition.<br>**Result**: The properties dialog box for the transition is opened, see also *Properties dialog box for transitions, page 960*. |
| 2 | In the **Type of transition condition** area, activate the **Variable** option button, if it isn't already activated. |

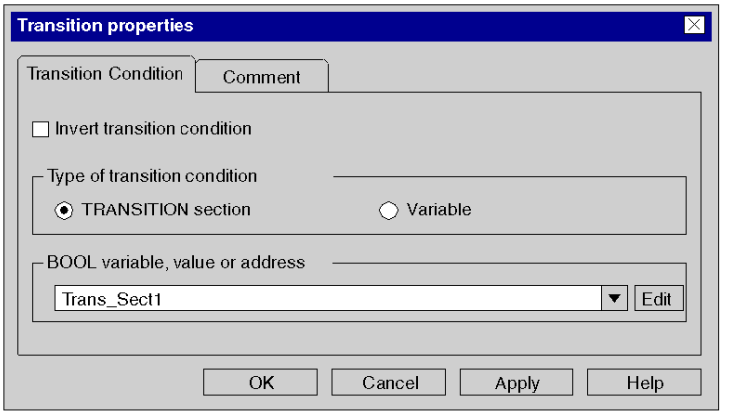
| Step | Action |
|---|---|
| 3 | In the **BOOL variable, value or address** data selection box *(see page 1911)*, enter a Boolean variable/address, a Boolean element for a multi-element variable or a Boolean literal, see also section Transition conditions *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>You have the following options to enter variables:<br>● You can enter the name of the variable.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Using the command button **...** you can open a variable selection dialog box *(see page 1182)* and confirm the selected variables there with **OK**.<br>Example:<br>  |

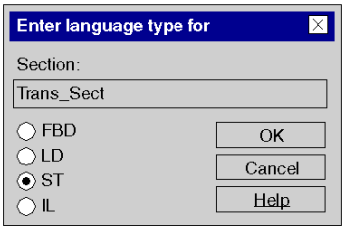
| Step | Action |
|---|---|
| 4 | Confirm the entry with **OK**.<br>**Result:**<br>● If the selected variable is already declared, the transition condition is accepted, the properties dialog box is opened, a syntax and semantics check is *(see page 867)* carried out and the procedure is ended.<br>  <br><br>● If the selected variable is not declared, a variable declaration dialog box opens. In this case, continue with the following steps of this procedure.<br>Variable declaration dialog box<br>  <br><br>**Note:** If you wish to declare the variable later, close the dialog box using the symbol. In this case, the variable name is enabled in the section, but not declared. You can declare the variables later in the Data Editor *(see page 370)* or by selecting the transition and the executing the menu command **Generate Variable** in the shortcut menu. |
| 5 | If you do not want to assign the variable an address or comment, confirm using **Enter** or the symbol.<br>**Result:**The variable is declared and enabled in the section. A syntax and semantics check *(see page 867)* is performed and the procedure ends here.<br>Example:<br>  <br> |
| 6 | To assign an address or comment, use the symbol.<br>**Result:** The dialog box expands.<br>  <br> |
| 7 | Enter the address or comment, or both. |

| Step | Action |
|---|---|
| 8 | Confirm with **Enter** or the ✓ symbol.<br>**Result:**The variable is declared and enabled in the section. A syntax and semantics check *(see page 867)* is performed.<br>Example:<br>  |

## Defining the transition condition as a section

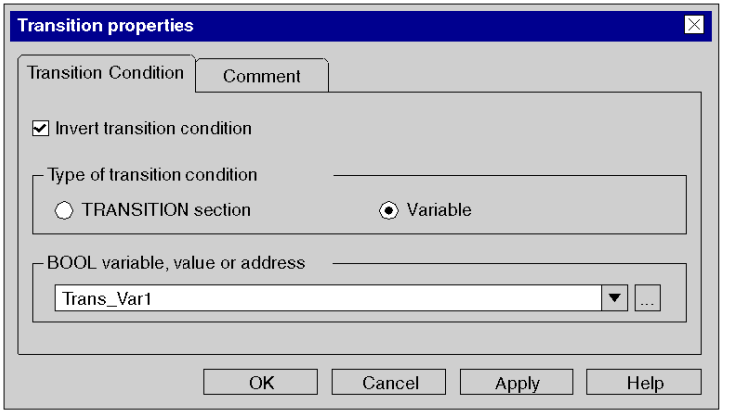
Defining the transition condition as a section:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the transition.<br>**Result**: The properties dialog box for the transition is opened, see also *Properties dialog box for transitions, page 960*. |
| 2 | In the **Type of transition condition** area, activate the **TRANSITIONS section** option button. |
| 3 | Enter the name of the transition section in the text box, see also section Transition section *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>You have the following options to enter the section name:<br>● You can enter the section name.<br>or<br>● You can select the name of the section from the list of transition sections which are defined but not yet used with the ▼ symbol.<br>Example:<br>  |

| Step | Action |
|---|---|
| 4 | To create the logic for the transition section, click the **Edit** command button.<br>**Result:** A dialog box to define the programming language of the transition section is opened.<br>Example:<br>  |
| 5 | Select the desired programming language for the transition section and confirm the selection with **OK**.<br>**Information for creating LD sections:** An LD section contains 11-64 columns and 17-2000 lines. The number of columns is set via the **Tools** → **Project Settings** dialog box, in the **Editors** tab in the **Number of columns** text box. These settings are only valid for new sections and not for existing sections. Always define the number of columns before you create a new LD section. (The number of lines can be increased or reduced during the creation of section contents.)<br>**Result:** The properties dialog box is closed and the transition section is opened. |
| 6 | Create the logic for the transition section, see also section Transition section *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |

## Inverting the transition condition

Inverting the transition condition:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the transition.<br>**Result**: The properties dialog box for the transition is opened, see also *Properties dialog box for transitions, page 960*. |
| 2 | To invert the transition condition, activate the **Invert transition condition** check box.<br>Example:<br>  |
| 3 | Confirm the entry with **OK**.<br>**Result:** The transition condition is represented as inverted in the section, the properties dialog box is closed and a syntax and semantics check *(see page 867)* is performed.<br>Example:<br>  |

## Enter a comment

Enter a comment:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the transition. |
| 2 | Select the **Comment** tab and enter a transition comment. |
| 3 | Confirm the comment with **OK**.<br>**Result:** The comment is accepted and the properties dialog box is closed. |

# Properties dialog box for transitions

## Calling the properties dialog box

see *Displaying the object properties, page 884*

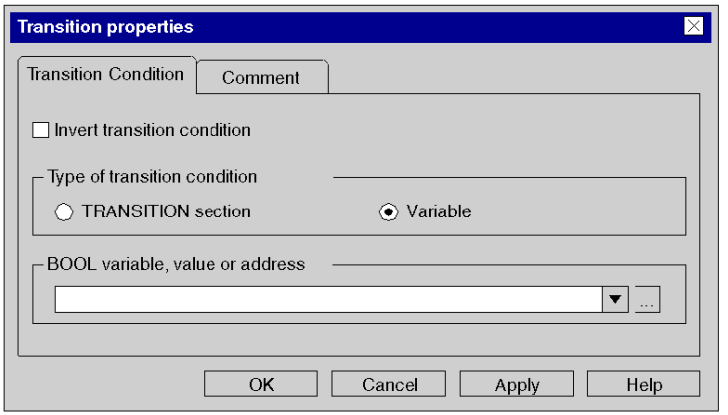
## Structure of the properties dialog box

The transition properties dialog box consists of two tabs:

- **Transition Condition**
  In this tab you can define:
  - the transition condition assigned to the transition as a transition variable *(see page 954)* or as a transition section *(see page 957)*
  - whether the transition condition should be inverted *(see page 959)*
- **Comment**
  In this tab, a comment *(see page 908)* about the transition can be entered.

## Transition Condition tab

Representation of the **Transition Condition** tab:

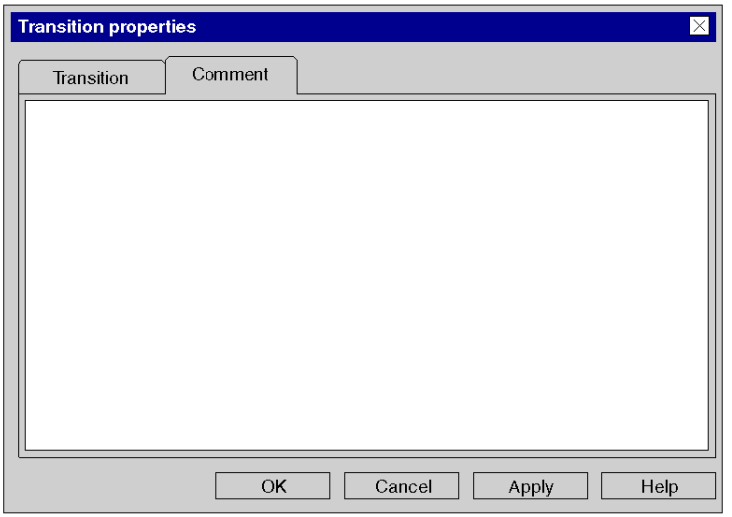


Elements of the **Transition Condition** tab:

| Element | Description |
|---|---|
| **Invert transition condition** | If you activate this check box, the transition condition is inverted (trigger the transition when 0). |
| **Type of transition condition** | In this area, you can define the transition condition, see also section Transition conditions *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>The contents of this area depends on the selected option:<br>● **Variable**<br>If you activate this option button, you can enter a Boolean variable/address, a Boolean element of a multi-element variable or a Boolean literal in the **BOOL variable, value or address** data selection box *(see page 1911)*.<br>● **TRANSITION section**<br>When you activate this option button, you can go to the **TRANSITION Section** text box and enter the name of the transition section.<br>**Note:** This text box is only used for referencing transition sections. If you want to rename an existing transition section, you must execute this in the properties dialog of the transitions section. |
| **Edit** | This command button is only available when the TRANSITION section option button is activated in the **Type of transition condition** area. Use this command button to close the properties dialog box and open the respective transition section. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the**Comment**tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the transition. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 23.9
# Editing Jumps

## Overview

This section describes how to edit jumps in the SFC programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Placing jumps | 964 |
| Defining jump properties | 966 |
| Jump properties dialog box | 968 |

# Placing jumps

## Introduction

Jumps are used to represent directional links which are not displayed in full length.

Jumps can be placed in any free cell.

If a jump is placed in a cell that is already occupied by an object, an error message is returned.

If a jump is placed in a cell where the neighboring cell above already contains a step, an error message is returned because a transition must be placed between a step and a jump (see also section Connection rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual).

A jump which is placed always connects automatically with the neighboring objects above if there are no free cells between them.

Use the context sensitive **Go to**, **Go to step** or **List and display jumps** dialogs to find a certain jump in the current section.

## Activating the jump placement mode

You have the following options to activate the jump placement mode:

- Use the **Edit** → **New** → **Jump** menu command.
- Use the **Jump** menu command from the shortcut menu.

  or

- Choose the  symbol.

The active jump placement mode is indicated by the  cursor symbol.

## Note

If in the **Tools** → **Options** dialog in the **Data and Languages** tab the **Automatically assign a variable to a new graphics object** checkbox is enabled, an objects property dialog will automatically open during placement. The procedures explained here are used when the check box is **cleared**.

## Placing jumps

Placing jumps:

| Step | Action |
|---|---|
| 1 | Activate the jump placement mode. |
| 2 | Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The jump is inserted and the selection mode is enabled again. |
| 3 | To insert further jumps:<br>● Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Defining jump properties

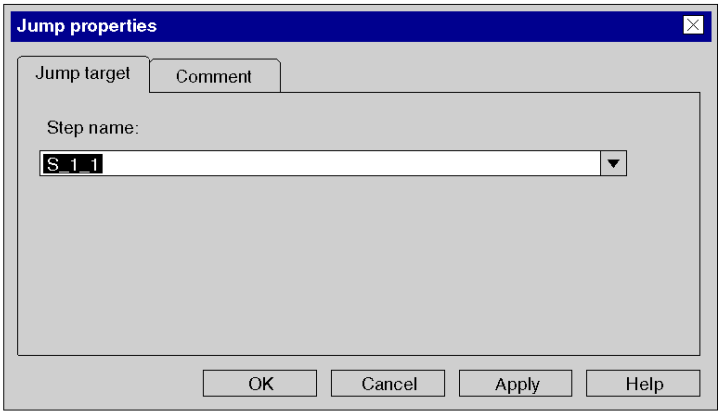
## Introduction

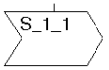
The following can be defined for jumps:

- the jump target *(see page 966)*, see also Jump *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
- a comment *(see page 967)*

## Defining the jump target

Defining the jump target:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the jump.<br>**Result**: The jump properties dialog box is opened, see also Calling the properties dialog box *(see page 968)*. |
| 2 | Enter the name of the target step in the text box, see also Jump *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>**Note:** With multi-token, you can enable the option to jump into a parallel sequence or out of a parallel sequence, see also the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab, **Sequential control (SFC)** area.<br>You have the following options to enter the target step name:<br>● You can enter the step name.<br>or<br>● You can select the name of the step from the list of steps used in the current section with the ▼ symbol.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Confirm the selection with **OK**.<br>**Result:** The properties dialog box is closed, the jump target is accepted and a syntax and semantics check *(see page 867)* is performed.<br>Example:<br>  |
| 4 | Create the logic for the transition section, see also Transition section *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |

## Entering a comment

Entering a comment:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the jump. |
| 2 | Select the **Comment** tab and enter a jump comment. |
| 3 | Confirm the comment with **OK**.<br>**Result:** The comment is enabled and the properties dialog box is closed. |

# Jump properties dialog box

## Calling the properties dialog box

You have the following options to call the properties dialog box:

- Double-click the jump.
- Select the jump and use the **Edit** → **Properties...** menu command.
  or
- Select the jump and use the **Object Properties...** menu command from the shortcut menu.

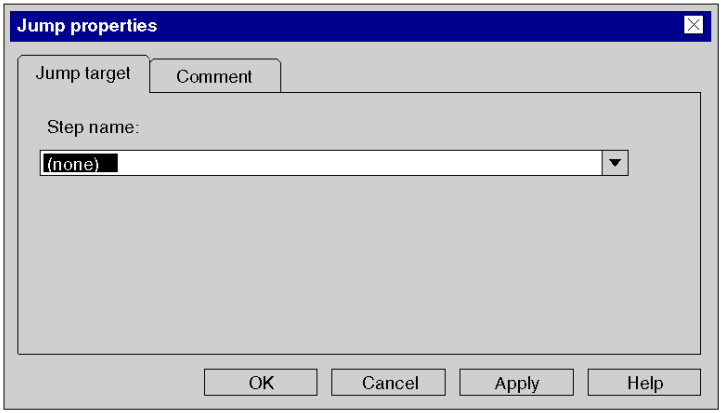
## Structure of the properties dialog box

The jump properties dialog box consists of two tabs:

- **Jump target**
  You can define the Jump target *(see page 966)* this tab.
- **Comment**
  In this tab, a comment *(see page 908)* about the transition can be entered.

## Jump target tab

Representation of the **Jump target** tab:

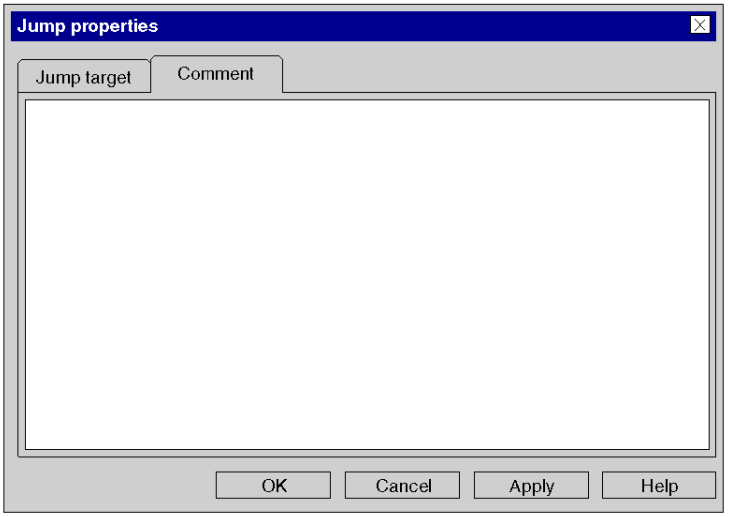



Elements of the **Jump target** tab:

| Element | Description |
|---|---|
| **Step Name** | Enter the name of the target step *(see page 966)* in this text box, see also section Jumps *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

## Comment tab

Representation of the **Comment** tab:




Elements of the **Comment** tab:

| Element | Description |
|---|---|
| Text box | Enter a comment about the transition. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 23.10
# Editing Branches and Joints

## Overview

This section describes how to edit branches and joints in the SFC programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selecting Branches and Joints | 971 |
| Placing branches and joints | 973 |
| Modifying the Properties of Branches and Joints | 974 |
| Branches properties dialog box | 977 |
| Joints properties dialog box | 978 |

# Selecting Branches and Joints

## Introduction

Alternative sequences and parallel sequences are differentiated in SFC:

- **Alternative sequence**
  The **alternative sequence** offers the option to program conditional branches in the control flow of the SFC structure, see also Alternative Branches and Alternative Joints *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
  Alternative sequences consist of:
  - **Alternative branch**
    With alternative branches, as many transitions follow a step under the horizontal line as there are different processes.
  - **Alternative joint**
    All alternative branches are linked to a single branch using alternative joints or jumps where they are then continued to be processed.
- **Parallel sequence**
  With parallel sequences the connection of an individual transition leads to the parallel activation of several (maximum 32) steps (branches). After this activation, the individual branches are processed independently from one another, see also Parallel Branches and Parallel Joints *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.
  Parallel sequences consist of:
  - **Parallel branch**
    With parallel branches, as many transitions follow a step under the horizontal line as there are parallel processes.
  - **Parallel joint**
    In accordance to IEC 61131-1, all parallel branches are joined at their ends by a common parallel joint. The transition to a parallel connection is evaluated when all previous steps of the transition are set.
    If a parallel branch is combined using an alternative joint, this is only possible with Multi-Token-Operation, see also dialog box **Tools → Project Settings** tab **Language extensions** area **Sequential control (SFC)**.

## Selecting Branches and Joints

Selecting Branches and Joints:

| Type | Call via menu command | Call via symbol | Call via keyboard | Cursor symbol |
|---|---|---|---|---|
| Alternative branch *(see page 973)* | **Edit → New → Alternative branch** or **Branches and Joints → Alternative branch** shortcut menu |  | **F4** |  |
| Alternative joint *(see page 973)* | **Edit → New → Alternative joint** or **Branches and Joints → Alternative joint** shortcut menu |  | **Shift+F4** |  |
| Parallel branch *(see page 973)* | **Edit → New → Parallel branch** or **Branches and Joints → Parallel branch** shortcut menu |  | **F5** |  |
| Parallel joint *(see page 973)* | **Edit → New → Parallel joint** or **Branches and Joints → Parallel joint** shortcut menu |  | **Shift+F5** |  |

# Placing branches and joints

## Introduction

Branches and joints do not require their own cell, but are inserted into the respective cell of the step or transition.

If a branch or joint is placed in a cell that already contains a branch or joint an error message is returned (see also section Connection rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual).

A branch or joint which is placed connects automatically with the neighboring objects above and below if there are no free cells between them.

## Placing branches or joints

Placing branches or joints:

| Step | Action |
|---|---|
| 1 | Select the desired branch or joint, see also *Selecting Branches and Joints, page 971*. |
| 2 | Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The selected branch or joint is inserted and the selection mode is active again. |
| 3 | To place further elements of the same type:<br>● Click the target cell in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Modifying the Properties of Branches and Joints

## Introduction

The following can be defined for branches:

- the number of output pins
  - by dragging the mouse *(see page 974)*
  - using the properties dialog box *(see page 975)*
- the position of the input pin
  - using the properties dialog box *(see page 976)*

The following can be defined for joints:

- the number of input pins
  - by dragging the mouse *(see page 974)*
  - using the properties dialog box *(see page 975)*
- the position of the output pins
  - using the properties dialog box *(see page 976)*

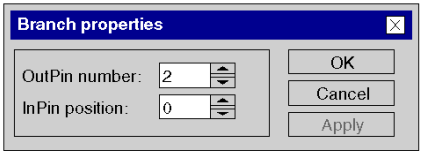
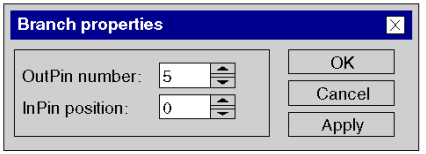
## Defining the number of pins using the mouse

Defining the number of pins using the mouse:

| Step | Action |
|---|---|
| 1 | Select the branch or joint.<br>**Result:** A change label is placed at both ends of the branch or joint.<br>Example:<br>  |
| 2 | Place the mouse pointer on one of the change labels.<br>**Result:** The mouse pointer changes its symbol to  . |
| 3 | Drag the branch or joint in the desired direction using the mouse.<br>**Result:** The branch or joint is expanded to the desired number of pins.<br>Example:<br>  |

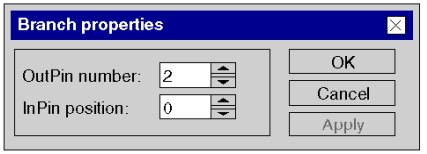
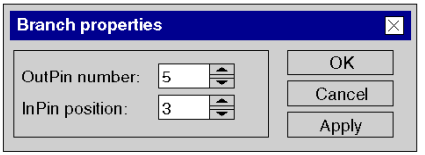
## Defining the number of pins using the properties dialog box

Defining the number of pins using the properties dialog box:

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the branch or joint.<br>**Result**: The properties dialog box for the branch *(see page 977)* or joint *(see page 978)* is opened.<br>Example of a properties dialog box for branches:<br>  |
| 2 | Enter the desired number of output pins (for branches) or the desired number of input pins (for joints).<br>Example of a properties dialog box for branches:<br>  |
| 3 | Confirm the entry with **OK**.<br>**Result:** The number of pins is accepted.<br>Example:<br>  |

## Defining the position of the input or output pin

Defining the position of the input pin (for branches) or output pin (for joints):

| Step | Action |
|---|---|
| 1 | Open the properties dialog box *(see page 884)* for the branch or joint.<br>**Result**: The properties dialog box for the branch *(see page 977)* or joint *(see page 978)* is opened.<br>Example of a properties dialog box for branches:<br>  |
| 2 | Enter the target position of the input pin (for branches) or the position of the output pin (for joints).<br>**Note:** The position numbers start at 0.<br>Example of a properties dialog box for branches:<br>  |
| 3 | Confirm the entry with **OK**.<br>**Result:** The position of the input or output pin is accepted.<br>Example:<br>  |

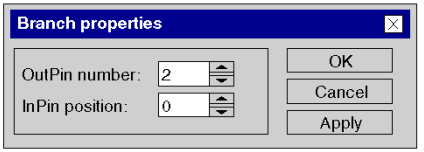
# Branches properties dialog box

## Calling the properties dialog box

see *Displaying the object properties, page 884*

## Build

Representation of the dialog box:




Elements of the dialog box:

| Element | Description |
|---|---|
| **OutPin number** | Enter the number of output pins *(see page 975)* of the branch in this text box. |
| **InPin position** | Enter the position of the input pin *(see page 976)* of the branch in this text box.<br>**Note:** The position numbers start at 0. |
| **OK** | Use this command button to accept all entries and close the properties dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

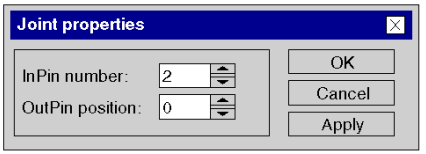
# Joints properties dialog box

## Calling the properties dialog box

see *Displaying the object properties, page 884*

## Build

Representation of the dialog box:




Elements of the dialog box:

| Element | Description |
|---|---|
| **InPin number** | Enter the number of input pins *(see page 975)* of the branch in this text box. |
| **OutPin position** | Enter the position of the output pin *(see page 976)* of the branch in this text box.<br>**Note:** The position numbers start at 0. |
| **OK** | Use this command button to accept all entries and close the dialog box. |
| **Apply** | Use this command button to accept all entries without closing the properties dialog box. |

# Section 23.11
# Inserting Alternative Strings and Simultaneous Strings

## Overview

This section describes how to insert alternative strings and simultaneous strings in the SFC programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selecting Alternative Sequences and Parallel Sequences | 980 |
| Placing alternative sequences and parallel sequences | 981 |

# Selecting Alternative Sequences and Parallel Sequences

## Introduction

You can place alternative sequence and parallel sequence objects (branches, steps, transitions, joints) individually or place entire alternative sequences or parallel sequences. You can select any number of branches or transitions (with alternative sequences) or steps (with parallel sequences).

See also section Branches and Joints *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

## Selecting Sequences

Selecting alternative sequences and parallel sequences

| Sequence type | Call via menu command | Call via symbol | Cursor symbol |
|---|---|---|---|
| Alternative sequence *(see page 981)* | **Sequences → Alternative Sequence** shortcut menu |  |  |
| Parallel Sequence *(see page 982)* | **Sequences → Parallel Sequence** shortcut menu |  |  |

# Placing alternative sequences and parallel sequences

## Introduction

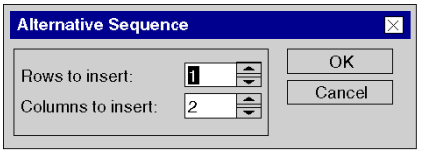
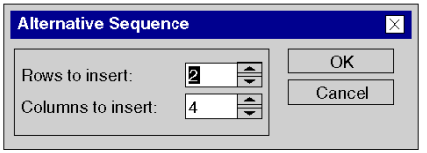
Alternative sequences and branches can only be placed in free cells. The number of free cells required depends on the size of the sequence to be inserted.

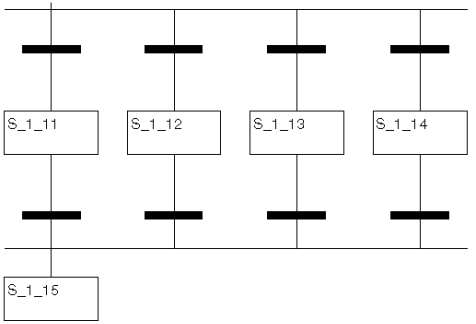
If a cell, in which an alternative sequence or parallel sequence is to be placed, is already occupied by an object, an error message is returned.

An alternative sequence or parallel sequence which is placed always connects automatically with the neighboring objects above and below if there are no free cells between them, see also section Connection rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

## Placing an alternative sequence

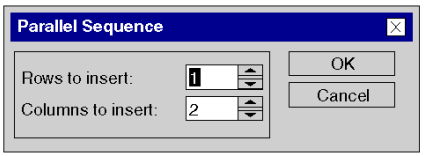
Placing an alternative sequence:

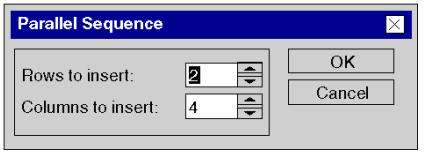
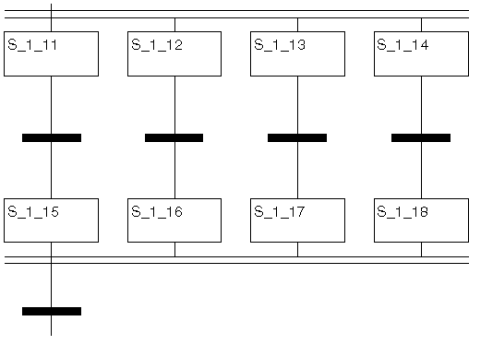
| Step | Action |
|---|---|
| 1 | Activate the placement mode for alternative sequences, see also *Selecting Alternative Sequences and Parallel Sequences, page 980*.<br>**Result:** The dialog box for defining the sequence size is opened.<br>Example:<br>  |
| 2 | Enter the number of transition-step sequences to be inserted per branch in the **Rows to insert** text box. |
| 3 | Enter the number of branches to be inserted into the **Columns to insert** text box.<br>Example:<br>  |
| 4 | Confirm the entries with **OK**.<br><br>**Result:** The dialog box is closed and the cursor changes its symbol to  . |

| Step | Action |
|---|---|
| 5 | Click the cell where the input pin of the alternative sequence should be placed.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The alternative sequence is inserted into the section.<br>Example:<br>  |
| 6 | Define the properties of the placed steps *(see page 902)* and transitions *(see page 954)*. |

## Placing a parallel sequence

Placing a parallel sequence:

| Step | Action |
|---|---|
| 1 | Activate the placement mode for parallel sequences, see also *Selecting Alternative Sequences and Parallel Sequences, page 980*.<br>**Result:** The dialog box for defining the sequence size is opened.<br>Example:<br>  |
| 2 | Enter the number of step-transition sequences to be inserted per branch in the **Rows to insert** text box. |

| Step | Action |
|---|---|
| 3 | Enter the number of branches to be inserted into the **Columns to insert** text box.<br>Example:<br>  |
| 4 | Confirm the entries with **OK**.<br><br>**Result:** The dialog box is closed and the cursor changes its symbol to  . |
| 5 | Click the cell where the input pin of the alternative sequence should be placed.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The parallel sequence is inserted into the section.<br>Example:<br>  |
| 6 | Define the properties of the placed steps *(see page 902)* and transitions *(see page 954)*. |

# Section 23.12
# Inserting Step-Transitions and Transition-Step-Sequences

## Overview

This section describes how to insert step transitions and transition step sequences using the SFC programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selecting Step-Transition and Transition-Step Sequences | 985 |
| Placing step-transition and transition-step sequences | 986 |

# Selecting Step-Transition and Transition-Step Sequences

## Introduction

You can place the steps and transitions individually or place complete Step-Transitions or Transition-Steps. Any number of steps and transitions can be selected.

## Selection

Selecting Step-Transitions and Transition-Step Sequences:

| Sequence type | Call via the menu command | Call via the icon | Cursor symbol |
|---|---|---|---|
| Insert Step-Transition-Sequence *(see page 986)* | **Sequence** → **Step-Transition**<br>**Sequence** shortcut menu |  |  |
| Transition-Step Sequence *(see page 987)* | **Sequence** → **Transition-Step**<br>**Sequence** shortcut menu |  |  |

# Placing step-transition and transition-step sequences

## Introduction

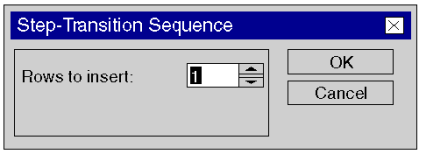
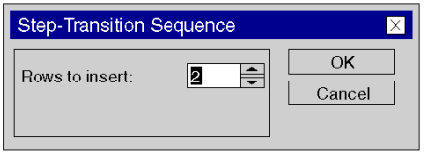
Step-transition and transition-step sequences can only be placed in free cells. The number of free cells required depends on the size of the sequence to be inserted.

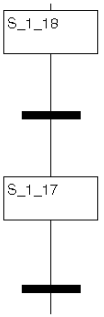
If a cell, in which the step-transition or transition-step sequence is to be placed, is already occupied by an object, an error message is returned.

A step-transition or transition-step sequence which is placed always connects automatically with the neighboring objects above and below if there are no free cells between them, see also section Connection rules *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

## Placing a step-transition sequence

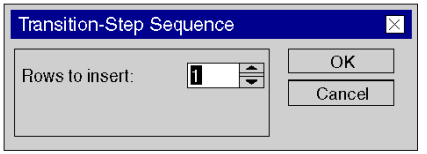
Placing a step-transition sequence:

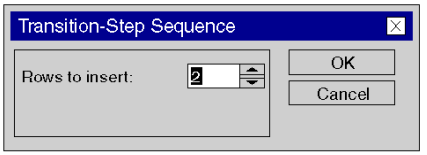
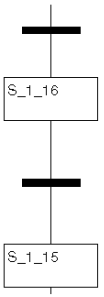
| Step | Action |
|---|---|
| 1 | Activate the placement mode for step-transition sequences, see also *Selecting Step-Transition and Transition-Step Sequences, page 985*.<br>**Result:** The dialog box for defining the number of step-transition sequences is opened.<br>Example:<br>  |
| 2 | Enter the number of step-transition sequences to be inserted into the **Rows to insert** text box.<br>Example:<br>  |
| 3 | Confirm the entry with **OK**.<br><br>**Result:** The dialog box is closed and the cursor changes its symbol to . |

| Step | Action |
|---|---|
| 4 | Click the cell where the input pin of the step-transition sequence should be placed.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The step-transition sequences are inserted into the section.<br>Example:<br>  |
| 5 | Define the properties of the placed steps *(see page 902)* and transitions *(see page 954)*. |

## Placing a transition-step sequence

Placing a transition-step sequence:

| Step | Action |
|---|---|
| 1 | Activate the placement mode for transition-step sequences, see also *Selecting Step-Transition and Transition-Step Sequences, page 985*.<br>**Result:** The dialog box for defining the number of transition-step sequences is opened.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | Enter the number of transition-step sequences to be inserted into the **Rows to insert** text box.<br>Example:<br>  |
| 3 | Confirm the entry with **OK**.<br><br>**Result:** The dialog box is closed and the cursor changes its symbol to . |
| 4 | Click the cell where the input pin of the transition-step sequence should be placed.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The transition-step sequences are inserted into the section.<br>Example:<br>  |
| 5 | Define the properties of the placed steps *(see page 902)* and transitions *(see page 954)*. |

# Section 23.13
# Editing Links

## Overview

Editing links in the SFC programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Links | 990 |
| Placing a Link | 991 |
| Editing links | 995 |

# Links

## Introduction

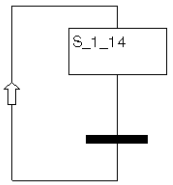
Links connect steps and transitions, transitions and steps, etc.

Attention should be paid to the following programming information:

- Links between objects of the same type (step with step, transition with transition, etc.) are not possible.
- Links are possible between:
  - Unused object outputs
  - Used or unused step inputs
    (i.e. step inputs can be connected using links)
- Overlapping links and other SFC objects (step, transition, jump, etc.) is not possible.
- Overlapping links and links is possible.
- Crossing links and links is possible. Crossed links are indicated by a "broken" link.



- The signal flow is generally from top to bottom. In exceptional cases (e.g. return) it can also be from bottom to top. This special case is indicated using an arrow symbol.




**NOTE:** A detailed description about links can be found in section Links *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.

## Selecting links

You have the following options to activate the placement mode:

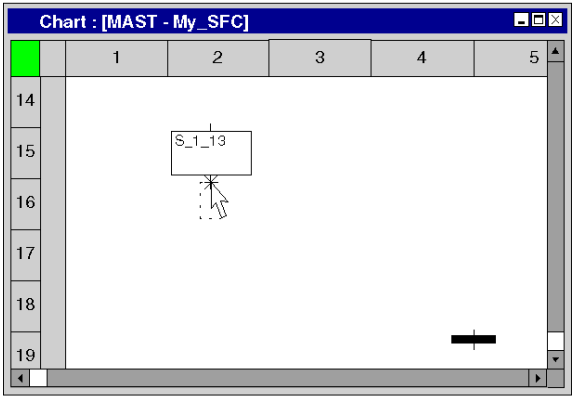
- Use the **Edit** → **New** → **Link** menu command.
- Use the **Link** menu command from the shortcut menu.
- Press the **F6** key.
  or
- Select the  symbol.

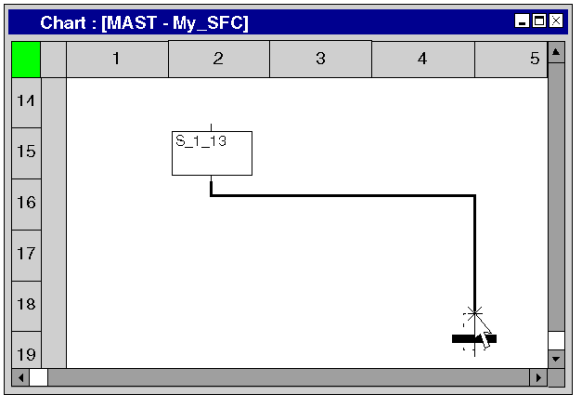
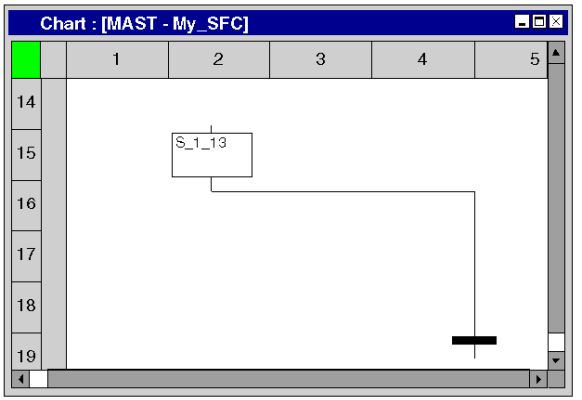
The  and  cursor symbols indicate which mode is active.

# Placing a Link

## Placing a link using the mouse

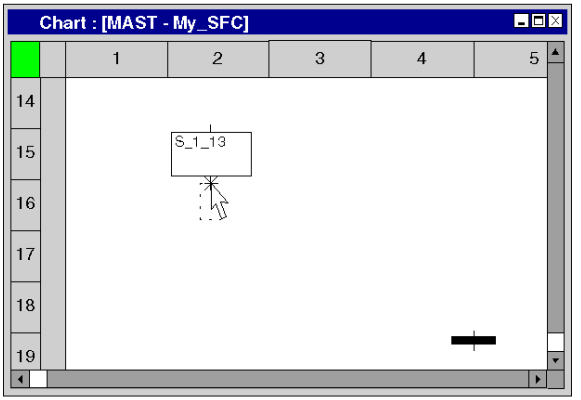
Carry out the following steps to place a link using the mouse:

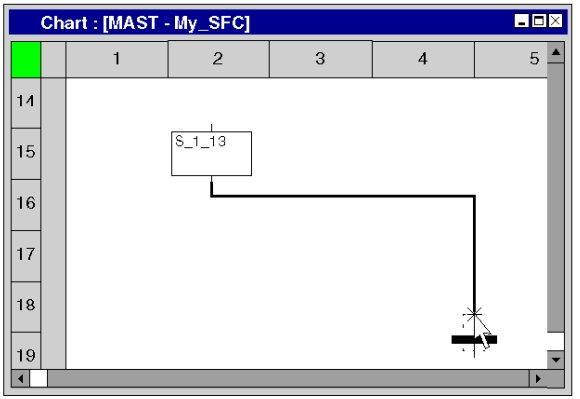
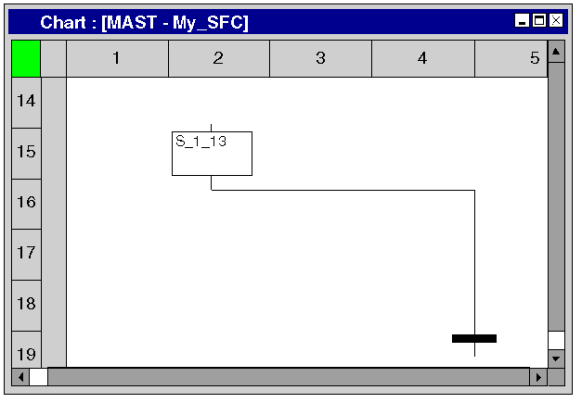
| Step | Action |
|---|---|
| 1 | Activate the placement mode for links, see also *Links, page 990*. |
| 2 | Place the mouse pointer on the start point of the link.<br>**Note:** The correct position is shown by the mouse pointer symbol.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Click the left mouse button on the start point and drag mouse pointer to the target point of the link.<br>Example:<br>  <br><br>**Tips:**<br>● By clicking between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● You can leave this mode at any time by pressing **Esc**. |
| 4 | Click on the target position with the left mouse button.<br>**Result:** The link is inserted.<br>Example:<br>  <br><br>See also *Editing links,* *page 995* |

## Placing a link using the keyboard

Carry out the following steps to place a link using the keyboard:

| Step | Action |
|---|---|
| 1 | Activate the placement mode for links, see also *Links, page 990*. |
| 2 | Use the **cursor keys** to move the cursor to the start point of the link, and press **Enter**.<br>**Note:** The correct position is shown by the cursor symbol.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Use the **cursor keys** to move the cursor to the target position of the link.<br>Example:<br>  <br><br>**Tips:**<br>● By pressing the **Enter** key between the start point and target position you can place intermediate points; for example, to avoid crossing other objects.<br>● You can leave this mode at any time by pressing **Esc**. |
| 4 | Press **Enter**.<br>**Result:** The link is inserted.<br>Example:<br>  <br><br>See also *Editing links, page 995* |

# Editing links

## Editing links

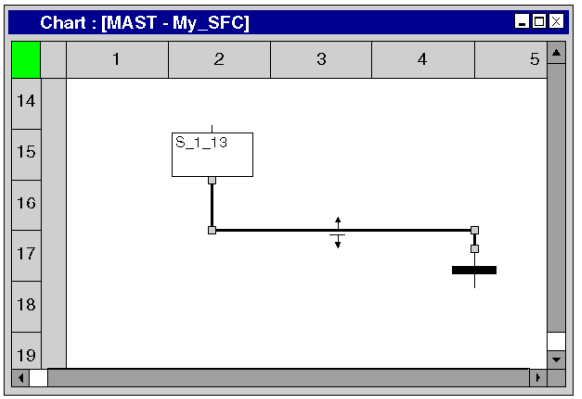
Links are comprised of one or more segments connected to one another.

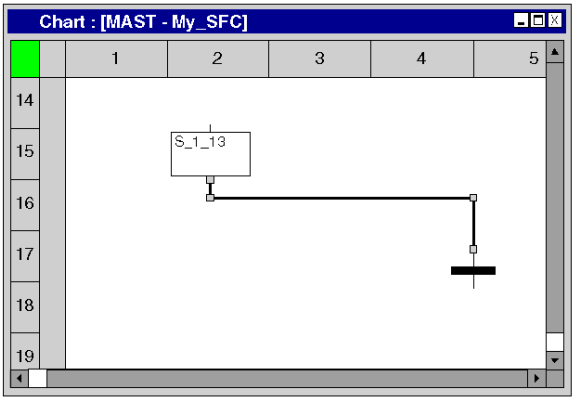
When cutting *(see page 875)*, deleting *(see page 874)*, copying *(see page 876)* and moving *(see page 878)* links, the action is executed for the entire link.

If a link is selected, the target points are displayed at their beginnings, ends and at every direction change

## Moving links vertically/horizontally

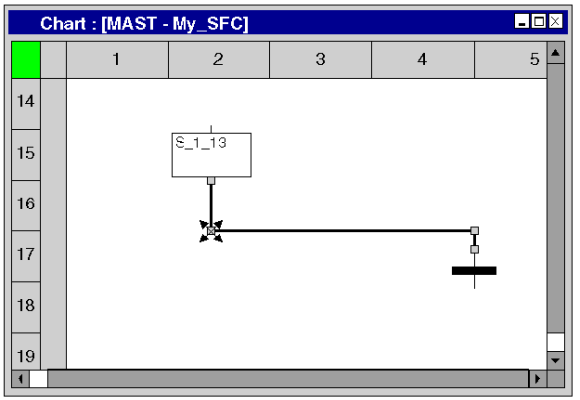
Process the following procedures to move a link segment vertically or horizontally:

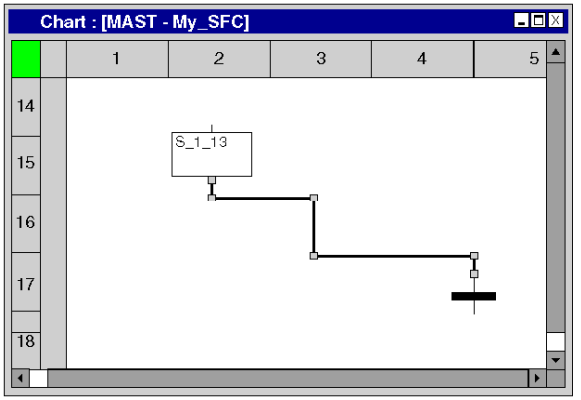
| Step | Action |
|---|---|
| 1 | Select the link. |
| 2 | Place the mouse pointer on the item to be moved.<br>**Note:** The correct position is indicated by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Click the left mouse button and drag the segment to the target position. |

| Step | Action |
|---|---|
| 4 | Release the mouse button.<br>**Result:**<br>The change is made.<br>  |

## Creating angles in FFB links

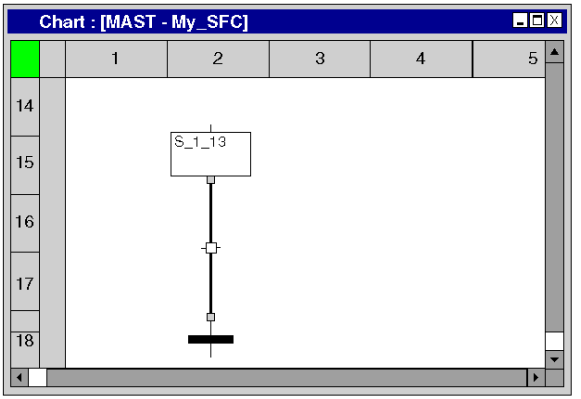
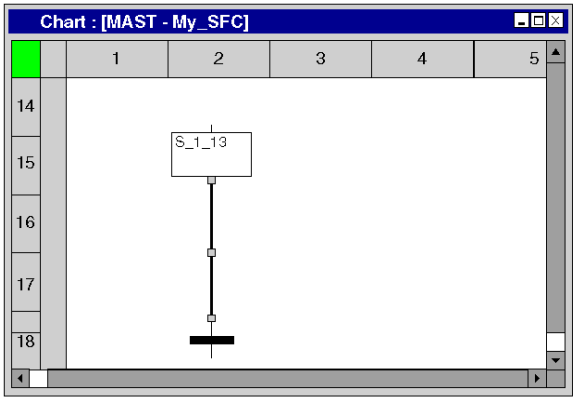
Process the following steps to create angles in links:

| Step | Action |
|---|---|
| 1 | Select the link. |
| 2 | Place the mouse pointer on a change label.<br>**Note:** The correct position is indicated by the mouse pointer symbol.<br>Example:<br>  |

| Step | Action |
|---|---|
| 3 | Click the left mouse button and drag the change label to the target position. |
| 4 | Release the mouse button.<br>**Result:**<br>The change is made.<br>  |

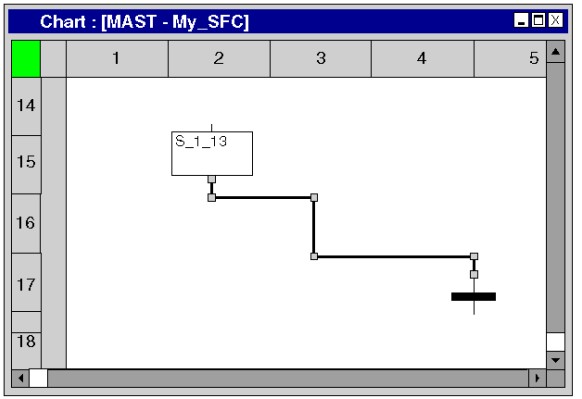
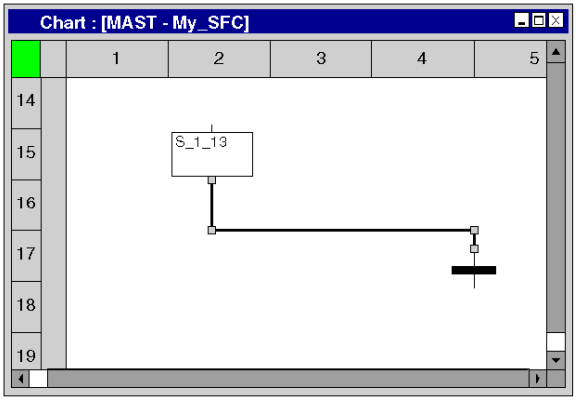
## Creating a sizing handle

The creation of angles in links can only be made using sizing handles. If no sizing handles are available they can be created. Process the following steps to create sizing handles in links:

| Step | Action |
|---|---|
| 1 | Select the link. |
| 2 | Press the **Ctrl** key and place the mouse pointer at the target position.<br>**Note:** The correct position is indicated by the mouse pointer symbol.<br>Example:<br>  |
| 3 | Left-click the target position.<br>**Result:**<br>The sizing handle is created.<br>  |

## Determining the optimum link route

Determining the optimum link route:

| Step | Action |
|---|---|
| 1 | Select the link.<br>Example:<br>  |
| 2 | Use the **Edit** → **Link** → **Route** menu command.<br>or<br>Use the **Route** menu command from the shortcut menu.<br>**Result:** The current link route is changed to the optimum route.<br>Example:<br>  |

# Section 23.14
# Entering Comments

## Entering comments

### Introduction

Comments can be placed in SFC in the form of text objects.

Text objects can overlap other objects.

The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill additional grid units.

The same rules apply for entering text and navigating within text objects as for editing ASCII text in standard text editors. (To create a line break in text objects you must press the **Ctrl**+**Enter** key combination).

**NOTE:** Keep in mind that each change to a comment (e.g. change to a comment text, change to the size of the text object) makes it necessary to recreate the section involved (**Generation** → **Generate project**).

### Selecting the text object

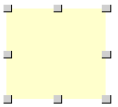
You have the following options to select text objects:

- Use the **Edit** → **New** → **Comment** menu command.
- Use the **Comment** menu command from the shortcut menu.
- Press the **F8** key.
  or
- Select the  symbol.

The active placement mode for text objects is indicated by the  cursor symbol.

## Placing text objects:

Placing text objects:

| Step | Action |
|---|---|
| 1 | Select the text object. |
| 2 | Click the target position in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>**Result:** The selected text object is inserted and the selection mode is active again.<br>Example:<br>  |
| 3 | Enter the comment. |
| 4 | Confirm the text entered by:<br>● Clicking with the mouse outside of the text object<br>or<br>● Pressing the **Enter** key. |
| 5 | To insert additional text objects:<br>● Click the target position in the SFC section.<br>or<br>Use the **arrow keys** to move the cursor to the target position, and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Section 23.15
# Online functions

## Online functions

### Online functions

The description of online functions is found in chapter *Debugging in Sequential Function Chart Language (SFC)*, *page 1455*.

# Section 23.16
# Export/Import

## Export/Import SFC Sections

### Export/Import

The description for exporting/importing sections is found in the chapter *Import / Export, page 1719*.

# Section 23.17
# Customize SFC Editor Colors

## Customize SFC Editor Colors

### Overview

To customize SFC editor colors, modify the SFCColors.ini in the Unity Pro folder.

For more detail on destination folder when Unity Pro is installed, refer to the *Unity Pro Installation Manual (see Unity Pro, Installation Manual)*.

**NOTE:** Make a copy of this file before modifying it.

The SFC editor colors are described in the table below.

### Description

This table describes the elements of the color file:

| Parameter | Default value | Description |
|---|---|---|
| BkAnimation | 192,192,192 | Editor background color when animation is ON and you are connected in programming mode. |
| StatementError | 255,0,0 | Element in error is underlined with this color. |
| InspectBk | 0,255,255 | Inspect window background color |
| InspectText | 0,0,0 | Inspect window text color |
| InspectBkMin | 255,255,0 | Inspect window background color when the value is less than the minimum value defined in the inspect window settings. |
| InspectBkMax | 255,0,255 | Inspect window background color when the value is greater than the maximum value defined in the inspect window settings. |
| BkAnimMonitoring | 255,211,211 | Editor background color when animation is ON and you are connected in monitoring mode. |
| AnalyzeError | 0,0,255 | Drawing color when an element generates an analyze error. |
| GraphAnimBoolTrue | 0,150,0 | Color of ANY_BOOL variable when its value is TRUE. |
| GraphAnimBoolFalse | 255,0,0 | Color of ANY_BOOL variable when its value is FALSE. |
| BkComment | 255,255,204 | Background color of comment area |
| BkCommentEdit | 225,225,225 | Background color of comment area when you are editing it. |
| BkAnimNumeric | 255,255,0 | Background color of numeric variables |
| BlockBreakpoint | 128,0,0 | Bullet color in the element where the breakpoint is set. |

| Parameter | Default value | Description |
|---|---|---|
| BlockCurrentStep | 255,255,0 | Color of the current element in debug mode |
| AnchorLink | 128,0,0 | Color of anchor link |
| BkAnimFrozen | 128,255,255 | Editor background color when animation is frozen (using SFC debug mode). |
| SFCStepTMax | 255,0,255 | Step color when the maximum time defined in the step is reached. |
| SFCStepTMin | 255,255,0 | Step color when the minimum time defined in the step is reached. |

# Chapter 24
# Common Function for Textual Language

## Overview

This chapter describes the common menus and dialog boxes of the IL and ST editor.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Selecting Text | 1008 |
| Deleting, cutting, copying, pasting and moving text | 1010 |
| Undoing and redoing changes | 1012 |
| Match case | 1013 |
| Zoom (font size) | 1014 |
| Using Bookmarks | 1015 |
| Go to | 1017 |
| Go to dialog box | 1019 |
| Searching text | 1022 |
| Searching and replacing text | 1024 |
| Call a subroutine | 1026 |
| Refine DFBs and subroutines | 1028 |
| Inspect windows | 1030 |
| Enter comments | 1034 |
| Printing the Current Section | 1036 |
| Initialize search | 1037 |
| Initialize Animation table | 1038 |
| Displaying the section properties | 1039 |
| Fullscreen | 1040 |

# Selecting Text

## Selecting one or more characters

Selecting one or more characters:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Left click with mouse and keep it pressed.<br>**2.** Drag the mouse across the characters you want to select.<br><br>or<br>**1.** Place the insertion point in front of the first character you want to select.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Click the left mouse button behind the last character you want to select. | **1.** Press the **Shift** key and keep it pressed.<br>**2.** Use the arrow keys to move the insertion point over the characters to be selected. |

## Selecting a word

Selecting a word:

| Using the mouse | Using the keyboard |
|---|---|
| Double click on the word to be selected. | **1.** Press the **Shift** key and keep it pressed.<br>**2.** Use the arrow keys to move the insertion point over the word to be selected. |

## Selecting one or more lines

Selecting one or more lines:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Click the left mouse button at the height of the line to be selected in the left margin.<br>or<br>**1.** Click the left mouse button at the height of the first line to be selected in the left margin.<br>**2.** Press and hold the mouse button down and drag the cursor down to the last line to be selected.<br>or<br>**1.** Place the insertion point at the start of the first line you want to select.<br>**2.** Left click the mouse and keep it pressed.<br>**3.** Drag the cursor to the end of the last line to be selected.<br>**4.** Release the mouse key.<br>or<br>**1.** Place the insertion point at the start of the first line you want to select.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Click the left mouse button behind the last line you want to select. | **1.** Press the **Shift** key and keep it pressed.<br>**2.** Use the arrow keys to move the insertion point over the characters to be selected. |

## Select all (entire contents of the section)

Select all (entire contents of the section):

| Using the mouse | Using the keyboard |
|---|---|
| Use the menu command **Edit** → **Select all**.<br>or<br>Press the **Ctrl** key, and left mouse click in the left margin. | Press the key combination **Ctrl**+**A**. |

# Deleting, cutting, copying, pasting and moving text

## Deleting Text

Deleting text:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 1008)* the text to be deleted.<br>2. Use the **Edit** → **Delete** menu command. | 1. Select *(see page 1008)* the text to be deleted.<br>2. Press the **Delete** key. |

## Cutting Text

Cutting text:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 1008)* the text to be cut.<br>2. Use the **Edit** → **Cut** menu command.<br>or<br>Use the **Cut** menu command from the shortcut menu (right-click).<br>or<br>Click the  symbol. | 1. Select *(see page 1008)* the text to be cut.<br>2. Press the **Ctrl**+**X** key combination. |

The text which is cut can be pasted *(see page 1011)* at any location (also in another IL or ST section).

## Copying Text into the Clipboard

Copying text into the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| 1. Select *(see page 1008)* the text to be copied.<br>2. Use the **Edit** → **Copy** menu command.<br>or<br>Use the **Copy** menu command from the shortcut menu (right-click).<br>or<br>Click the  symbol. | 1. Select *(see page 1008)* the text to be copied.<br>2. Press the **Ctrl**+**C** key combination. |

The text which is copied can be pasted *(see page 1011)* at any location (also in another IL or ST section).

## Pasting Text from the Clipboard

Pasting text from the clipboard:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Place the cursor at the target position.<br>**2.** Use the menu command **Edit** → **Insert**.<br>or<br>Use the menu command **Paste** in the shortcut menu (right mouse button).<br>or<br>Click the  symbol. | **1.** Place the cursor at the target position.<br>**2.** Press the **Ctrl**+**V** key combination. |

## Moving Text

Moving text:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 1008)* the text to be moved.<br>**2.** Position the mouse pointer on the selected text.<br>**3.** Click the left mouse button and keep it pressed.<br>**4.** Drag the text to the new position.<br>**Note:** This can also be done between sections in various open IL and ST sections.<br>**5.** Release the mouse button.<br>**Result:** The selected text is moved from the original position to the target position. | - |

## Copying text

Copying text:

| Using the mouse | Using the keyboard |
|---|---|
| **1.** Select *(see page 1008)* the text to be moved.<br>**2.** Position the mouse pointer on the selected text.<br>**3.** Click the left mouse button and keep it pressed.<br>**4.** Press the **Ctrl** key and keep it pressed.<br>**5.** Drag the text to the new position.<br>**Note:** This can also be done between sections in various open IL and ST sections.<br>**6.** Release the mouse button first and then the **Ctrl** key.<br>**Result:** A copy of the selected text is pasted at the target position. | - |

# Undoing and redoing changes

## Undo changes

You have the following options to undo the recent changes:

- Use the **Edit** → **Undo** menu command.
- Press the **Ctrl**+**Z** key combination.
  or
- Select the  symbol.

One change is undone each time the function is executed.

The function can be executed a maximum of 10 times.

## Redo changes

Changes that have been undone using **Undo** can be restored:

- Use the **Edit** → **Redo** menu command.
- Press the **Ctrl**+**Y** key combination.
  or
- Select the  symbol.

One undone change is restored each time the function is executed.

The function can be executed a maximum of 10 times.

# Match case

## All caps text

Process the following steps to display text in All Caps:

| Step | Action |
|---|---|
| 1 | Select the text to display in All Caps. |
| 2 | You have the following options to convert the selected text:<br>● Use the **Edit** → **All caps** menu command.<br>● Press the **Shift**+**Alt**+**U** key combination.<br>● Select the  symbol. |

## Lower case text

Process the following steps to display text in lower case:

| Step | Action |
|---|---|
| 1 | Select the text to display in lower case. |
| 2 | You have the following options to convert the selected text:<br>● Use the **Edit** → **Lower case** menu command.<br>● Press the **Alt**+**U** key combination.<br>● Select the  symbol. |

# Zoom (font size)

## Introduction

Several font sizes are available to provide an optimal display of the text in the current section (overview of the entire contents or display of details).

## Zoom out (reducing the font size)

To reduce current display (font size) by 25 %:

- Select the menu **View** → **Zoom** → **Zoom out**.
- Right click in the section and select **Zoom** → **Zoom out** for the section.
- Press the key combination **Ctrl**+**Minus (-)**.
- Click the  symbol.
- Click the  symbol.
- Using the mouse: press **Ctrl**+**Rotate the mouse wheel**.

## Zoom in (increasing the font size)

To increase current display (font size) by 25 %:

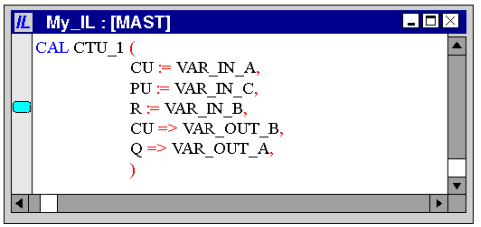
- Select the menu **View** → **Zoom** → **Zoom in**.
- Right click in the section and select **Zoom** → **Zoom in** for the section.
- Press the key combination **Ctrl**+**Plus (+)**.
- Click the  symbol.
- Click the  symbol.
- Using the mouse: press **Ctrl**+**Rotate the mouse wheel**.

# Using Bookmarks

## Introduction

Bookmarks are used to mark text lines and allow these marked lines to be easily found in different sections.

An example of a line marked with a bookmark:




## Setting a Single Bookmark

You have the following options to set a bookmark:

- Place the cursor in the line to be marked and carry out the menu command **Edit → Bookmarks> → Toggle Bookmark**.
- Place the cursor in the line to be marked and press the key combination **Ctrl+F2**.
  or
- Place the mouse pointer in the margin (gray column on the left edge of the section) for the line to be marked and use **Toggle Bookmark** from the context menu of the margin.

## Setting Multiple Bookmarks

Setting a bookmark in each line which contains a defined search text can be done in the following ways:

- Use the menu command **Edit → Find...** to open the search dialog box, enter the search text and press the command button **Mark all**.
  or
- Press the key combination **Ctrl+F** to open the search dialog box, enter the search text and press the command button **Mark all**.

Also see *Searching text, page 1022*

### Removing Single Bookmarks

You have the following options to remove a bookmark:

- Place the cursor in the line with the bookmark to be removed and carry out the menu command **Edit → Bookmark> → Toggle Bookmark**.
- Place the cursor in the line with the bookmark to be removed and press the key combination **Ctrl+F2**.
  or
- Place the mouse pointer on the bookmark to be removed and use **Toggle Bookmark** from the context menu of the bookmark.

### Removing All Bookmarks

You have the following options to remove all bookmarks in the current section:

- Use the **Edit → Bookmarks → Clear all Bookmarks** menu command.
  or
- Press the **Ctrl+Shift+F2** key combination.

### Searching for Bookmarks (Search Down)

You have the following options to search for bookmarks (search down):

- Use the **Edit → Bookmarks → Next Bookmark** menu command.
- Use the **Edit → Go to → Bookmark → Next** menu command.
  or
- Press the **F2** key.

The bookmarks are traversed according to their order of appearance in the **Bookmarks** dialog.

### Searching for Bookmarks (Search Up)

You have the following options to search for bookmarks (search up):

- Use the **Edit → Bookmarks → Previous Bookmark** menu command.
- Use the **Edit → Go to → Bookmark → Previous** menu command.
  or
- Press the **Shift+F2** key combination.

The bookmarks are traversed according to their order of appearance in the **Bookmarks** dialog.

### Bookmarks Dialog

Please also refer to Bookmarks Dialog *(see page 578)*.

# Go to

## Introduction

The **Go to** function is used to jump to:
- a specific line in the current section,
- a specific bookmark in the current section or
- a specific mark in the current section,

## Open the dialog

The following options are available to open the**Go to**dialog box.
- Use the menu command **Edit** → **Go to**,
- select the menu command **Go to** from the sections shortcut menu.
- Press the key combination **Ctrl**+**G**.
  or
- Choose the  symbol.

## Go to a line

Go to a certain line in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 1017)* the **Go to**dialog box. |
| 2 | Select the **Line** tab. |
| 3 | Enter the line number. |
| 4 | Confirm the entry using the command button **Go to**.<br>**Result:** The desired line is displayed. |

## Go to a bookmark

Go to a certain bookmark in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 1017)* the **Go to**dialog box. |
| 2 | Choose the **Bookmark** tab. |
| 3 | Select a bookmark in the list. |
| 4 | Double click on the selected bookmark or press the **Go to**command button.<br>**Result:** The desired bookmark is displayed. |

## Go to a location

Go to a certain location in the current section by carrying out the following steps:

| Step | Action |
|---|---|
| 1 | Open *(see page 1017)* the **Go to**dialog box. |
| 2 | Select the **Location** tab. |
| 3 | Select a location in the list. |
| 4 | Double click on the selected location or press the **Go to**command button.<br>**Result:** The desired location is displayed. |

# Go to dialog box

## Call the dialog box:

see *Open the dialog, page 1017*

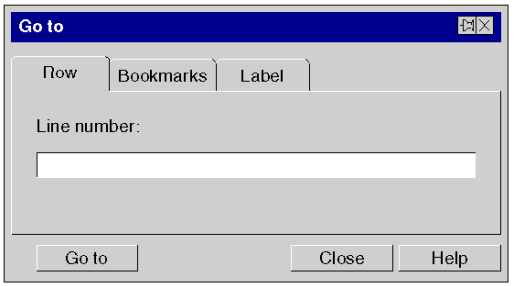
## Call the dialog box

The **Go to**dialog box is made up of three tabs:

- **Row**
  Go to a certain location in the current line using this tab.
- **Bookmarks**
  Go to a certain bookmark in the current section using this tab:
- **Label**
  Go to a certain location in the current section using this tab:

## tab Line

Representation of the**Line** tab:

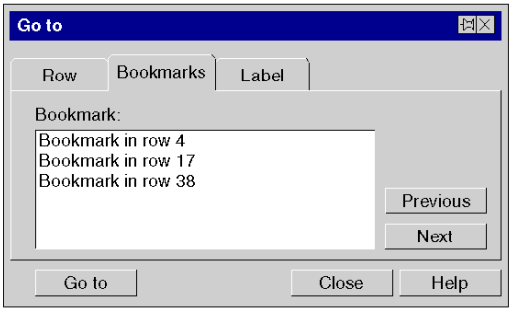



Elements of the tab **Line**:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Line number** | The number of the row to be displayed can be entered in this text box. |
| **Go to** | Display the current position using the button desired line *(see page 1017)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

## Register tab Bookmarks

Representation of the**Bookmark** tab:

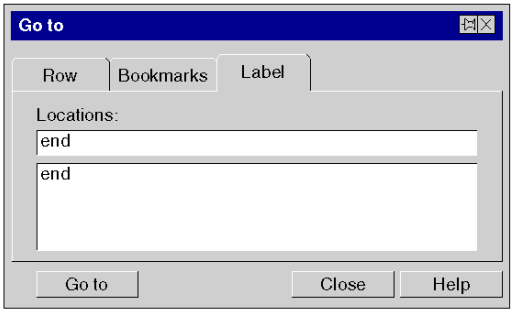



Elements of the tab **Bookmark**:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Bookmark:** | Select the bookmark to be displayed from this list box. |
| **Prev** | Display the current position using this button.<br>When the start of the section is reached, the search starts again from the bottom. |
| **Next** | Display the next bookmark using this button.<br>When the end of the section is reached, the search starts again from the top. |
| **Go to** | Display the current position using the button desired bookmark *(see page 1017)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

## Register tab Location

Representation of the tab **Label**:




Elements of the tab **Bookmark**:

| Element | Description |
|---|---|
|  | For symbols that are not printed, the dialog box is closed after every search.<br>For printed symbols the dialog box remains open until you close it. |
| **Locations:** | Select the location to be displayed from this text box.<br>You can enter the name of the location in the text box with the keyboard or using the drop-down list by clicking with the mouse. |
| **Go to** | Display the current position using the button desired location *(see page 1018)*. |
| **Close** | Use this button to close the dialog. |
| **Help** | Use this button to call up the dialog help. |

# Searching text

## Searching text

When searching text, you can define the search text and various options for the search.

You have the following options to search for text:

- Use the **Edit** → **Find** menu command.
- Press the key combination **Ctrl**+**F**.
  or
- click the  symbol.

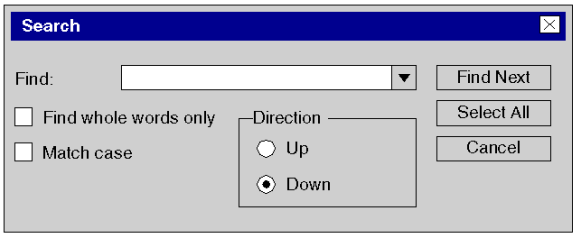
This function is only used for searching and in some cases marking search terms.

To search and replace text, use the **Edit** → **Replace** menu command, see also *Searching and replacing text, page 1024*.

For intelligent search and replace of variables and FFBs you can use the **Tools** → **Search/Replace** menu command.

## Representation

Search dialog display




Elements of the dialog box

| Element | Description |
|---|---|
| **Find:** | Enter the text to be found. You can insert text from the clipboard or from the list of recently used terms. |
| **Find whole words only** | If you select this option, only the whole term is searched for and not parts of the term. |
| **Match case** | If you select this option, only terms are searched whose capitalization matches the text in the **Find what** text box. |
| **Direction** | Select the desired direction here. |
| **Find Next** | Use this command button to search for the next instance of the search target in the section. |
| **Select All** | Use this command button to mark all lines in the section that contain the search term with a bookmark, see also *Using Bookmarks, page 1015*. |

### Find Previous

You have the following options to find earlier instances of the term defined in the Find dialog box:

- Use the **Edit** → **Previous** menu command.
  or
- press the **Shift**+**F3** key combination.

### Find Next

You have the following options to find later instances of the term defined in the Find dialog box:

- Use the **Edit** → **Find Next** menu command.
  or
- press the **F3** key.

# Searching and replacing text

## Searching and replacing text

You have the following options to search and replace text:

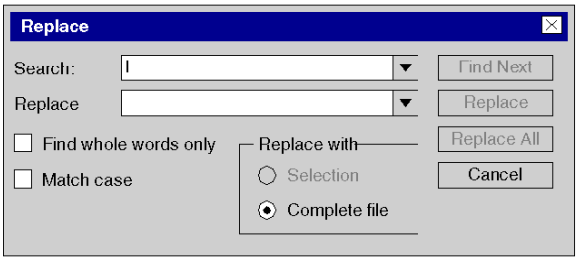
- Use the **Edit** → **Replace** menu command.
  or
- Press the **Ctrl**+**H** key combination.

This function is used to search and replace simple text.

For intelligent search and replace of variables and FFBs you can use the **Tools** → **Search/Replace** menu command.

## Representation

Representation of the Find dialog box




Elements of the dialog box

| Element | Description |
|---|---|
| **Find what** | Enter the text to be found. You can insert text from the clipboard or from the list of recently used terms. |
| **Replace with** | Enter the new text. You can insert text from the clipboard or from the list of recently used terms. In order to delete the text in the **Find what** text box, leave the **Replace with** text box empty. |
| **Find whole words only** | If you select this option, only the whole term is searched for and not parts of the term. |
| **Match case** | If you select this option, only terms are searched whose capitalization matches the text in the **Find what** text box. |
| **Replace with** | Select the area where the replace operation should be performed:<br>● **Selection**<br>The replace procedure is performed in the text area selected in the section.<br>● **Complete file**<br>The replace procedure is performed in the entire section. |

| Element | Description |
|---|---|
| **Find Next** | Use this command button to search for the next instance of the search target in the section. (No replacement is made.) |
| **Replace** | Use this command button to replace the selected occurrence of the search term and select the next occurrence. |
| **Replace All** | Use this command button to replace all occurrences of the search term in the area defined in **Replace with** with the replacement term. |

# Call a subroutine

## Introduction

The subroutine to be called must be located in the same task as the section calling.

Subroutines can also be called from within subroutines.

Subroutine calls are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow subroutines** check box.

In IL, a subroutine call consists of the `CAL` operator, followed by the name of the subroutine section, followed by an empty parameter list (optional), e.g.

```
CAL SR_Name
```

or

```
CAL SR_Name ()
```

In ST, a subroutine call consists of the name of the subroutine section, followed by an empty parameter list, followed by a semicolon, e.g.

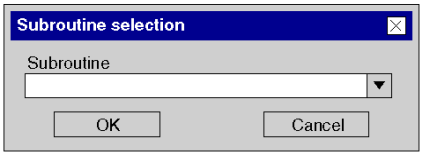
```
SR_Name () ;
```

The name of subroutine is shown in boldface to identify that these subroutines can be refined *(see page 1028)*.

You can enter the name of the subroutine section just like any other text, or use the input assistant.

## Entering the name of the subroutine section with input assistant

Process the following steps to enter a subroutine section name with the input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | Use the **Edit** → **New** → **Select subroutine** menu command.<br>or<br>Select the  symbol.<br>**Result:**<br>A selection dialog box for selecting the subroutine to be called is opened.<br>  |

| Step | Action |
|---|---|
| 3 | Click the ▾ symbol, select the subroutine to be called from the list and confirm pressing **Enter**.<br>**Note:** The subroutine to be called must be located in the same task as the section calling.<br>**Result:** The name of the subroutine section is enabled in the section. |
| 4 | Complete the subroutine call.<br>IL example:<br>`CAL MY_SR`<br>ST example:<br>`MY_SR () ;` |

# Refine DFBs and subroutines

## Introduction

This function makes it possible to "look into" a DFB or subroutine. This makes it possible to see the internal state of the DFB/subroutine when animation is used, or to make changes to the logic very quickly.

Refinable objects are labeled in boldface, e.g.

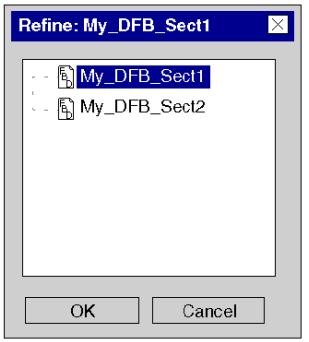
```
CAL SR_Name
```

or

```
MY_DFB (IN1 := var1, IN2 := var12, OUT => var3);
```

## Refining DFBs

Process the following steps to refine DFBs:

| Step | Action |
|---|---|
| 1 | Place the cursor on the DFB instance name. |
| 2 | You have the following options to refine DFBs:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl+Q** key combination.<br>**Result:** If the DFB contains only one section, this will be opened automatically. If the DFB contains several sections, a selection dialog is opened.<br>Example:<br>  |
| 3 | Select the desired DFB section.<br>**Result:** The DFB section is opened.<br>**Note:** With nested DFBs several **refine** processes are possible. |

## Refining subroutines

Process the following steps to refine subroutines:

| Step | Action |
|---|---|
| 1 | Position the cursor on the subroutine call. |
| 2 | You have the following options to refine subroutines:<br>● Use the **Services** → **Refine** menu command.<br>● Use the **Refine** menu command from the shortcut menu.<br>or<br>● Press the **Ctrl**+**Q** key combination.<br>**Result:** The subroutine is opened.<br>**Note:** With nested subroutines several **refine** processes are possible. |

# Inspect windows

## Introduction

In offline mode, the name, address (if available) and the comment (if available) of the associated variable are displayed in inspect windows.

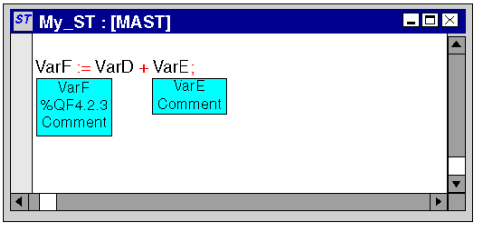
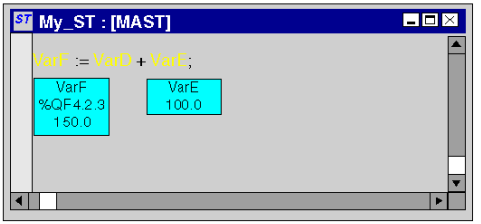
In online mode, the name, address (if available) and the current value of the variable are displayed in inspect windows.

The current values can be displayed:

- Decimal, e.g. `15`
- Binary, e.g. `00001111`
- Hexadecimal, e.g. `F`
- Floating point, e.g. `15.4`
- Exponent, e.g. `1.3e+003`
- Literal, e.g. `'otto'`

## Generating an inspect window

Generating an inspect window:

| Step | Action |
|---|---|
| 1 | Position the cursor in front of or in the target variable. |
| 2 | You have the following options to generate inspect windows:<br>● Use the **Edit** → **New** → **Inspect window** menu command.<br>● Use the **Place Inspect** menu command from the shortcut menu.<br>● Press the **F9** key.<br>or<br>● Select the  symbol.<br>**Result:** The inspect window is generated.<br>Example (offline):<br>  <br>Example (online):<br>  |

## Resizing

Position the mouse pointer on the right or left edge of an inspect window (the correct position is indicated by the ↔ or ↔ mouse pointer symbol), click the left mouse button and drag the mouse right or left.

The size selected applies to all inspect windows within the current section.

### Changing the position

Position the mouse pointer on the inspect window, click the left mouse button and move the inspect window to the target position.

### Anchoring the position

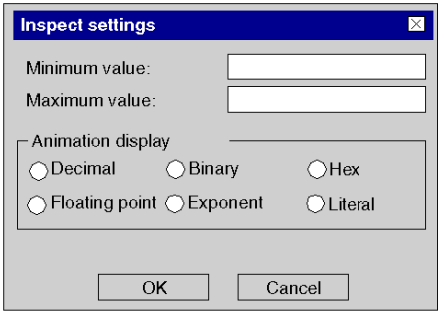
You can anchor all inspect windows of a section to their relative positions within the section (the inspect window stays in its onscreen position even if the page is scrolled), or to their absolute position (the inspect window is scrolled along with the rest of the section contents).

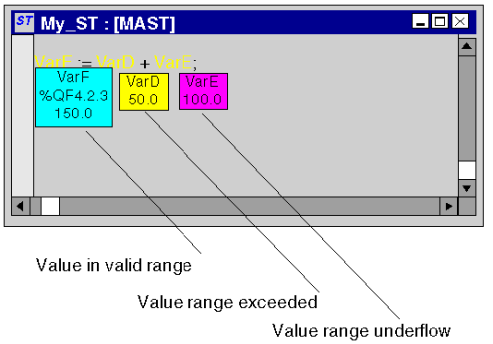
To switch between modes, select the **View → Floating Inspect Windows** menu command.

You can see which mode is active by way of a check-symbol in front of the menu command.

### Setting a zone monitor

Setting a zone monitor:

| Step | Action |
|---|---|
| 1 | Position the mouse pointer on the inspect window. |
| 2 | Use the **Settings** menu command from the shortcut menu.<br>**Result:** The properties dialog box for the inspect window is opened.<br>Example:<br>  |
| 3 | Enter a minimum and a maximum value for the variable to be monitored. |
| 4 | Enter the format of the variable displayed in the section. |

| Step | Action |
|---|---|
| 5 | Confirm the entries with **OK**.<br>**Result:**<br>● If the current value is within the defined range, the inspect window is displayed in cyan.<br>● If the current value is outside the defined range, the inspect window is displayed in yellow.<br>● If the current value is above the defined range, the inspect window is displayed in magenta.<br>Example:<br>  <br> |

## Hiding inspect windows

To hide and show the inspect windows select the **View** → **Hide Inspect Windows** menu command.

You can see which mode is active (hide inspect windows) by way of a check-symbol in front of the menu command.

## Deleting inspect windows

Position the cursor on the inspect window to be deleted and select the **Delete Inspect** menu command from the shortcut menu.

# Enter comments

## Introduction

In the IL and ST editors, comments start with the string `(*` and end with the string `*)`. Any comment can be entered between these two strings.

**NOTE:** Keep in mind that each change to a comment (e.g. change to a comment text, change to the size of the text object) makes it necessary to recreate the section involved (**Generation** → **Generate project**).

In accordance with IEC 61131-3, nesting of comments is not permitted. If it is necessary to nest comments you must enable them via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow nested comments** check box.

## Placing a new comment

Placing a comment:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | You have the following options for placing a comment:<br>● Use the **Edit** → **New** → **Comment** menu command.<br>● Use the **Comment** menu command from the shortcut menu.<br>● Press the **F8** key.<br>or<br>● Select the  symbol.<br>**Result:** The strings for the start and end of the comments are inserted.<br>Example:<br>`(*  *)` |
| 3 | Enter the comment between the two strings.<br>Example:<br>`(* Comment *)` |

## Executing Text Comments

Placing a comment:

| Step | Action |
|---|---|
| 1 | Select the command text to be executed.<br>Example:<br>`CAL My_Count`<br>`(CU:=Var1,R:=Var2,`<br>`PV:=Var3,Q=>Var4)` |
| 2 | You have the following options for placing a comment:<br>● Use the **Edit** → **New** → **Comment** menu command.<br>● Use the **Comment** menu command from the shortcut menu.<br>● Press the **F8** key.<br>or<br>● Select the  symbol.<br>**Result:** The strings for the start and end of the comments are inserted.<br>Example:<br>`(*CAL My_Count`<br>`(CU:=Var1,R:=Var2,`<br>`PV:=Var3,Q=>Var4)*)` |

# Printing the Current Section

## Introduction

There are the following ways to print the current section:

- Print the section using the menu command
- Print the section using the print documentation

## Print the section using the menu command

Carry out the following steps to print the current section using the menu command:

| Step | Action |
|---|---|
| 1 | Open the print dialog using:<br>● the menu command **File** → **Print**,<br>● or the key combination **Ctrl**+**P**<br>or<br>● the symbol  . |
| 2 | Press **OK** to start printing the current section. |

## Print the section using the print documentation

Carry out the following steps to print the current section using the print documentation:

| Step | Action |
|---|---|
| 1 | Double-click on **Documentation** in the project browser.<br>**Result:** An overview of your system is open, see also *Documentation, page 1695* |
| 2 | Select **Program** → **Tasks** → **MAST/FAST/AUX** → **...** to print the section. |
| 3 | Select the menu command **Print** from the sections shortcut menu. |
| 4 | Press **OK** to start printing the current section. |

# Initialize search

## Introduction

The **Initialize Search** function transfers the name of the selected element (search text) to the **Element** text box in the **Cross-References** window.

## Selecting the search text

Process the following steps to select the search text:

| Step | Action |
|---|---|
| 1 | Select the element to search for (search text).<br>The following elements can be searched for:<br>● Variable<br>(To search for a variable, select the variable.)<br>● EFB/DFB Instance<br>(To search for an EFB/DFB instance, select the name of the EFB/DFB instance.) |
| 2 | Use the **Services** → **Initialize Search** menu command.<br>or<br>Select the menu command **Initialize search** from the sections shortcut menu.<br>or<br>Press the **Ctrl**+**U** key combination.<br>**Result:** The search text is transferred to the **Element** text box in the **Cross-References** window. |

# Initialize Animation table

## Introduction

The **Initialize Animation Table** transfers the selected variables to the Animation table *(see page 1502)*.

## Selecting the variables

Process the following steps to select variables:

| Step | Action |
|---|---|
| 1 | Select the variables to transfer to the Animation table.<br>The following items can be selected:<br>● single variable<br>(To accept a variable, select the variable.)<br>● several variables<br>(To accept several variables, select the entire text area in which the variables are located.) |
| 2 | Use the **Services** → **Initialize Animation table** menu command.<br>or<br>Select the menu command **Animation Table** from the sections context menu.<br>or<br>Press the **Ctrl**+**T** key combination.<br>**Result:** The variable(s) is/are transferred to the Animation table *(see page 1502)*. |

## Initialize Animation Table

Using **Initialize Animation Table** (**Ctrl**+**T**) you create an animation table with a default name e.g. `Table[ST Editor - Mixer : [MAST]]`. You can rename the table.

All the variables you selected are listed in the animation table.

Selecting a new set of variables and using **Initialize Animation Table** again, all new variables are added to this animation table (if you did not rename it).

## Initialize New Animation Table

Using **Initialize New Animation Table** (**Ctrl**+**Shift**+**T**) you create a new animation table with an indexed default name e.g. `Table[ST Editor - Mixer : [MAST]1]`. You can rename the table.

All the variables you selected are listed in the table.

But if you click anywhere in the section and do not select any variable, you create a new empty animation table.

Every time you use **Initialize New Animation Table** you create a new animation table.

# Displaying the section properties

## View section properties

You have the following options to display the section properties dialog box *(see page 461)*:

- Use the **Edit → Data Selection...menu command**.
- Use the **Properties** from the shortcut menu.
- Press the **Alt**+**Enter** key combination.

## View Data Properties

You have the following options to display the section properties dialog box *(see page 1197)*:

- using the data editor *(see page 325)*
  **a.** Selecting one or several lines in a list editor
  **b.** Using the shortcut menu select the **Properties** command.
- via the IL/ST section.
  **a.** Selecting one or several element(s) in the IL/ST section.
  **b.** From the context menu select the command **Data properties** or press **Ctrl** + **Enter**.

# Fullscreen

## Fullscreen

You have the following options to display the section application window in full screen mode:

- Select the **View** → **Full screen** menu.
- Press the key combination **Ctrl**-**F8**.
- Click the  symbol.

To end the full screen mode press **Esc** or use the menu.

# Chapter 25
# IL Editor

## Overview

This chapter describes the specific menus and dialog boxes of the IL editor.

For a description of the syntax for the IL programming language, please refer to the Instruction List IL *(see Unity Pro, Program Languages and Structure, Reference Manual )* chapter in the Reference manual.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Structure of an IL program | 1042 |
| Creating an IL Program | 1044 |
| Syntax and Semantics Check during Programming | 1045 |
| Navigating with the keyboard | 1049 |
| Data input | 1050 |
| Calling an FFB | 1057 |
| Using Public Variables | 1068 |
| Online functions | 1070 |
| Reference Data Type in IL | 1071 |
| Export/Import IL Sections | 1072 |

# Structure of an IL program

## Introduction

An IL program (Instruction List) consists of a list of statements that are executed in sequence by the controller. Using these statements you can call function blocks, functions and procedures conditionally or unconditionally, process assignments and jumps conditionally or unconditionally within a section.

## Properties of an IL program

Properties of an IL program:

- Statements consist of:
  - an operator,
  - with modifier if necessary,
  - an operand if required and
  - a comment if required.
  - In addition, every statement can be provided with a label.
- Every statement begins on a new line.
- Each line is limited to 300 characters.
- Line breaks are possible in statements (multi-line statements).
- Labels, symbols and comments can be placed anywhere in a section. (Comments can be entered anywhere where empty spaces are permitted.)
- A syntax and semantics check *(see page 1045)* is performed immediately after the statement is entered. The result of this check is displayed in colored text.
- Syntactically or semantically incorrect sections can be saved.

## Edit and view functions

Edit and view functions of the IL editor:

- Entering text using the insert or overwrite mode *(see page 1044)*
- Deleting text *(see page 1010)*
- Selecting text *(see page 1008)* (characters, words, lines, all)
- Cutting *(see page 1010)*, copying *(see page 1010)* and pasting *(see page 1011)* text
- Moving text *(see page 1011)* (also between different ST/IL sections)
- Undo *(see page 1012)* and Redo *(see page 1012)*
- Go To function *(see page 1017)*
- Using bookmarks *(see page 1015)*
- Searching for text and marking the result *(see page 1022)*
- Searching and replacing text *(see page 1024)*
- Searching and replacing variables and function blocks and DFB instances
- Displaying text in selectable fonts and sizes
  The text display type can be selected from the **View** → **Select Font** menu command. This configuration applies to all sections of the IL and ST programming languages.
- Intelligent text indentation
  The text indentation of a new line is taken from the previous one.
- Display the position (line, column) of the cursor in the status bar.

## Input assistants

Input assistant for the IL editor:

- Input assistant for variables *(see page 1050)*
- Input assistant for functions, function blocks and procedures *(see page 1057)*

## Online functions

Online functions of the IL editor:

- Displaying actual values *(see page 1030)*.
- Setting breakpoints *(see page 1432)*
- Setting watchpoints *(see page 1437)*
- Step by Step *(see page 1434)*

# Creating an IL Program

## Writing in insert/overwrite mode

Text can be entered in insert or overwrite modes

- Insert mode
  The characters entered are inserted in the current position of the insert marker in addition to the existing characters.
- Overwrite mode
  The characters entered are inserted in the current position of the insert marker and overwrite the existing characters.

Switching between the modes is done using the **Insert** key.

The current mode is displayed in the status bar using **INS** for insert mode and **OVR** for overwrite mode.

## Creating an IL program

Carry out the following procedures to create an IL program:

| Step | Action |
|---|---|
| 1 | Creating an IL Section *(see page 458)*. |
| 2 | Enter the first operator (e.g. `LD`).<br>**Note:** As soon as the text is entered several checks are made, such as syntax/semantic error checks, correct spelling of keywords and variable names etc. A detailed description of syntax is found in chapter Instruction List IL *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>The results of the checks are indicated by a color folder, see also *Syntax and Semantics Check during Programming, page 1045*. |
| 3 | Use the **TAB** key or spaces to separate operands from operators. |
| 4 | Enter the operands (e.g. `il`). |
| 5 | Confirm the line with the **ENTER** key. |
| 6 | Repeat these steps until all instructions are entered.<br>Example:<br>`LD il`<br>`INT_TO_REAL`<br>`ADD r4`<br>`ST r3` |

# Syntax and Semantics Check during Programming

## Introduction

Syntax and semantics check is performed directly when creating the program.

The result of this check is displayed in three formats:

- directly in the program section with colored text
- as tooltip if the cursor is placed on incorrect text.
- In the output window, if **Generate** → **Analyze** is selected.

## Representation

Representation of colors and labels:

| Labeling | Description | Example |
|---|---|---|
| blue | Keyword (e.g. `LD`, `AND`, `ST`,...)<br>**Note:** Function block input operators (see also section Input operator requirements *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual) are not treated as keywords. | `LD i1` |
| red | Operators (that are not keywords) | `CAL My_Count`<br>`(CU:=Var1,R:=Var2,`<br>`PV:=Var3,Q=>Var4)` |
| green | Comment | `LD i1 (* Comment *)` |
| black | standard text (e.g. variable, address, label, ...) | `start: LD B` |
| bold | The name of DFBs and subroutines are shown in boldface to identify that these objects can be refined *(see page 1028)*. | `CAL My_DFB`<br>`(IN1:=Var1,`<br>`IN2:=Var2,`<br>`OUT=>Var3)`<br><br>`CAL MY_SUBROUTINE` |
| underlining | In DFB sections, the formal parameters of the DFBs are shown underlined. | `LD IN1`<br>`AND IN2`<br>`ST OUT` |

| Labeling | Description | Example |
|---|---|---|
| red wavy line | Faulty text:<br>● Syntax error<br>e.g. spelling errors in keywords, non-declared variables or FB instances, incorrect block call, unavailable block call (in the current PLC plateform or in the library types), EFB type call instead of the instance,...<br>● Semantic error<br>e.g. division by zero, value range exceeded/underflow (recognized by the system when entered) | LDs i1<br>CAL TON<br>(IN:=Var1,<br>PT:=Var2,<br>Q=>Var3<br>ET:=Var4) |
| Inverted text | Text is selected | LD :1 |

## Tooltips

If the cursor is placed over incorrect text, the tooltip is displayed with a brief description of the cause of the error. It will also be displayed in the output window after the analysis.

## Error messages in the output window

Error message: `Accessed object is not a function.`

| Cause of error | Error Correction | Example |
|---|---|---|
| Write error is present during accessing of a function. | Correct the write error. | Wrong:<br>LEQ (IN1 := VarE, IN2 := VarF)<br>ST VarG<br>Right:<br>LE (IN1 := VarE, IN2 := VarF)<br>ST VarG |
| Name of the function was already used for:<br>● one variable<br>● an FB instance<br>● a DFB Type<br>● an SR Section<br>● a procedure or function in a user library | Rename the already used object. | - |

| Cause of error | Error Correction | Example |
|---|---|---|
| Wrong syntax used during access<br>● an FB | Correct the syntax. | Wrong:<br>`My_CTD (`<br>`  CD := VarH,`<br>`  LD := VarI,`<br>`  PV := VarJ,`<br>`  Q  => VarK,`<br>`  CV => VarL)`<br>Right:<br>`CAL My_CTD (`<br>`  CD := VarH,`<br>`  LD := VarI,`<br>`  PV := VarJ,`<br>`  Q  => VarK,`<br>`  CV => VarL)`<br>Wrong:<br>`CAL TON`<br>`  (IN:=Var1,`<br>`  PT:=Var2,`<br>`  Q=>Var3`<br>`  ET:=Var4)`<br>Right:<br>`CAL My_TON`<br>`  (IN:=Var1,`<br>`  PT:=Var2,`<br>`  Q=>Var3`<br>`  ET:=Var4)` |

Error message: `Accessed object is not a function block.`

| Cause of error | Error Correction | Example |
|---|---|---|
| Write error is present during accessing.<br>● an FB | Correct the write error. | Wrong:<br>`CAL MyCTD (`<br>`  CD := VarH,`<br>`  LD := VarI,`<br>`  PV := VarJ,`<br>`  Q  => VarK,`<br>`  CV => VarL)`<br>Right:<br>`CAL My_CTD (`<br>`  CD := VarH,`<br>`  LD := VarI,`<br>`  PV := VarJ,`<br>`  Q  => VarK,`<br>`  CV => VarL)` |
| Name of the function was already used for:<br>● one variable<br>● an FB instance<br>● a DFB Type<br>● an SR Section<br>● a procedure or function in a user library | Rename the already used object. | - |
| Function block instances that are not defined in the data editor. | Define the FB on the data editor. | - |
| EFB that is not available for the current PLC platform. | Change the EFB. | - |
| An EFB that is not in the libset of the current project. | Update the libset. | - |

# Navigating with the keyboard

## Navigating with the keyboard

The following keys and key combinations are provided for navigation:

| Key combinations | Motion |
|---|---|
| **Cursor left** | Moves the cursor to the left |
| **Cursor right** | Moves the cursor to the right |
| **Cursor up** | Moves the cursor up |
| **Cursor down** | Moves the cursor down |
| **Ctrl**+**Cursor left** | Moves the cursor one string left |
| **Ctrl**+**Cursor right** | Moves the cursor one string right |
| **Ctrl**+**Cursor up** | Scrolls one line up (the position of the cursor is not affected by this function) |
| **Ctrl**+**Cursor down** | Scrolls one line down (the position of the cursor is not affected by this function) |
| **Home** | Moves the cursor to the beginning of the current line |
| **End** | Moves the cursor to the end of the current line |
| **Ctrl**+**Home** | Moves the cursor to the beginning of the current section |
| **Ctrl**+**End** | Moves the cursor to the end of the current section |
| **Page Up** | Scrolls one page up (the cursor remains in its position on the screen) |
| **Page Down** | Scrolls one page down (the cursor remains in its position on the screen) |
| **Ctrl**+**Alt**+**Page Up** | Displays the previous section (including via the **View** → **Previous Section**menu). |
| **Ctrl**+**Alt**+**Page Down** | Displays the next section (including via the **View** → **Next Section**menu). |
| **Alt**+**Enter** | Opens the properties dialog box for the selected section. |

# Data input

## Introduction

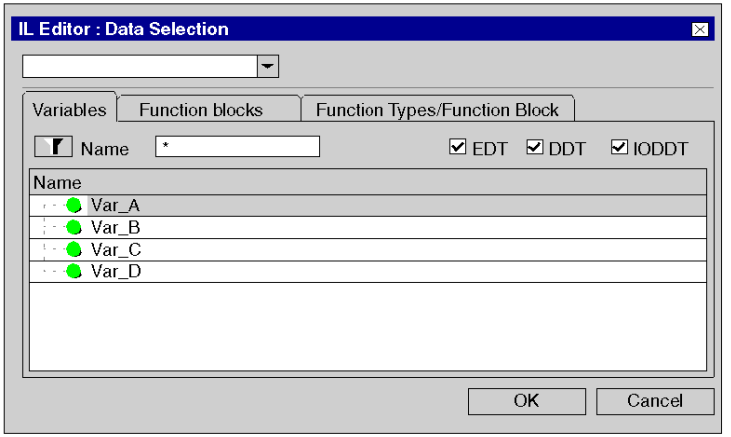
You have different options for using data in your IL section.

- Using already declared variables *(see page 1050)*
  - you can use a data selection dialog box,
    or
  - you can enter the variable name just like any other text.
- Use undeclared variables and declare them later *(see page 1052)*
  You can define variable names when creating your IL section and then declare all variables used.
- Use undeclared variables and declare them immediately *(see page 1052)*
  You can define variable names when creating your IL section and declare them immediately.
- Declaration of variables directly in the IL section. *(see page 1053)*
  You can declare variables in the Data Editor *(see page 370)*, but also directly in the IL section.
- Using formal parameters from function blocks (inputs/outputs, public variables) *(see page 1054)*
  - You can use a data selection dialog box.
    or
  - You can enter the formal parameter name just like any other text.

## Entering declared variables using data selection

Carry out the following steps to enter a declared variable using data selection:

| Step | Action |
|---|---|
| 1 | Declare the required variables, see also *Creation of EDT type variable instances, page 370*. |
| 2 | Place the cursor at the target position. |
| 3 | Open data selection by:<br>● The **Edit → Data Selection menu command.**.<br>● The **Data Selection... menu command**from the shortcut menu for the section.<br>or<br>● Press the **Ctrl**+**D** key combination.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |

| Step | Action |
|---|---|
| 4 | You have the following options to enter variables:<br>● You can now enter the variable name and confirm using **Enter** or the button ✓.<br>● You can select the variable names from the list of recently used names using the ▼ symbol and confirm using **Enter** or the button ✓.<br>● You can delete your entry by pressing **Esc** or the button ✗.<br>● Use the button **...**to open a variable selection dialog box *(see page 1189)* and in the **Variables** tab confirm the selected variable using the **OK** button.<br>Example of a data selection dialog box:<br>  <br><br>**Result:** The selected variable is inserted into the section and a syntax and semantics check *(see page 1045)* is performed. |

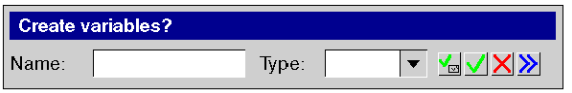
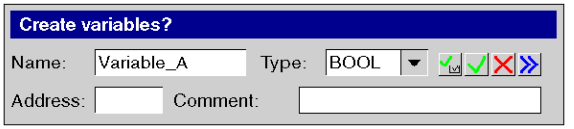
## Entering undeclared variables and declaring them later

Process the following steps to enter undeclared variables and declare them later:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | Enter the variable name.<br>**Result:** The variable selected is accepted. The syntax and semantics check *(see page 1045)* marks the variable name with a red wavy line to identify it as being undeclared, e.g.<br>VarA |
| 3 | Enter all variables in this way. |
| 4 | Declare all variables in the Data Editor *(see page 370)* or directly in the IL section *(see page 1053)*.<br>**Result:** The red wavy line under the variable name disappears. |

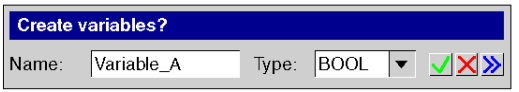
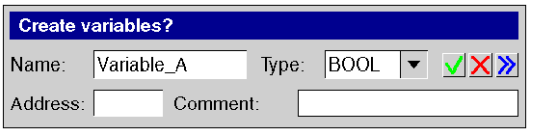
## Entering undeclared variables and declaring them immediately

Process the following steps to enter undeclared variables and declare them immediately:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | Click **Create Variable** in the shortcut menu, or press the **Shift**+**Enter** key combination<br>**Result:** The variable declaration dialog box opens.<br>  |
| 3 | Enter the variable name and data type. |
| 4 | If you do not want to assign an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and inserted at the target position in the section. |
| 5 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 6 | Confirm with **Enter** or the  symbol.<br>**Result:** The variable is declared and inserted at the target position in the section. |

## Declaration of variables directly in the IL section.

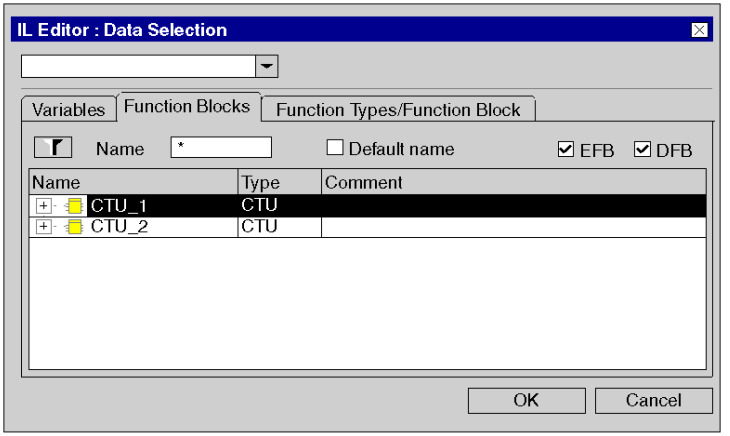
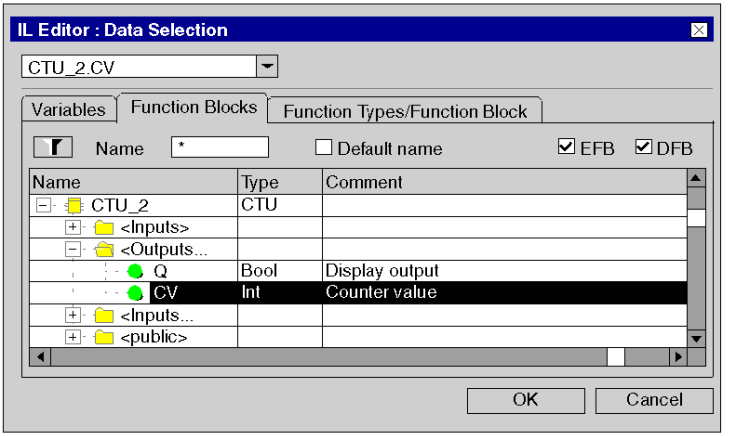
Process the following steps to declare variables directly in the IL section.

| Step | Action |
|---|---|
| 1 | Place the cursor in an undeclared variable (undeclared variables are identified by a red wavy line). |
| 2 | Click **Create Variable** in the shortcut menu, or press the **Shift**+**Enter** key combination<br>**Result:** The variable declaration dialog box opens.<br>  <br>**Note:** The variable name automatically appears in the **Name** field, if editor recognizes the text entered in the section as a variable and its syntax is correct. |
| 3 | If you do not want to assign an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears. |
| 4 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 5 | Confirm with **Enter** or the  symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears. |

## Using formal parameters

Process the following steps to use formal parameters with data selection:

| Step | Action |
|---|---|
| 1 | Call the function block whose formal parameters you want to use, see also Calling a function block *(see page 1057)*. |
| 2 | To use the formal parameter, place the cursor at the target position. |
| 3 | Open data selection by:<br>● The **Edit → Data Selection menu command.**.<br>● The **Data Selection... menu command**from the shortcut menu for the section.<br>or<br>● Press the **Ctrl**+**D** key combination.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |
| 4 | You have the following options to enter the formal parameter:<br>● You can enter the formal parameter name and confirm using **Enter** or the ✓ button.<br>In this case the procedure ends here.<br>● You can select the name from the list of recently used names using the ▼ symbol and confirm using **Enter** or the button ✓.<br>In this case the procedure ends here.<br>or<br>● Use the button **...** to open a data selection dialog box *(see page 1189)*. |

| Step | Action |
|---|---|
| 5 | Select the **Function Blocks** tab.<br>**Result:**<br>All declared block instances are displayed.<br>  |
| 6 | Expand the display of the inputs, outputs, and any inputs/outputs and public variables by clicking their **+** symbols.<br>**Result:**<br>You can now see all available inputs/outputs of the function block.<br>  |

| Step | Action |
|---|---|
| 7 | Select the desired formal parameter and confirm with **OK**.<br>**Result:** The selected formal parameter is inserted into the section and a syntax and semantics check *(see page 1045)* is performed.<br>For example:<br>`LD CTU_2.CV` |

# Calling an FFB

## Introduction

You have the following options to open FFBs (Elementary Function, Elementary Function Block, Derived Function Block, Procedure):

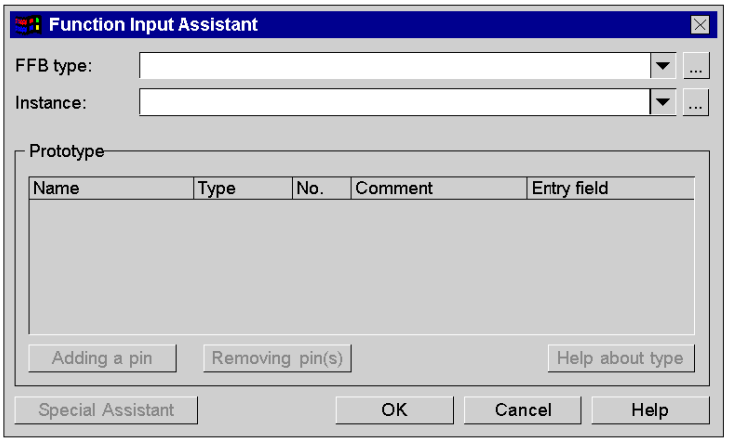
- You can enter the call like you enter any text (for syntax, see Calling FFB sections *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual),
- you can use the function input assistants,
  or
- you can enter the FFB to the section using the drag & drop method.

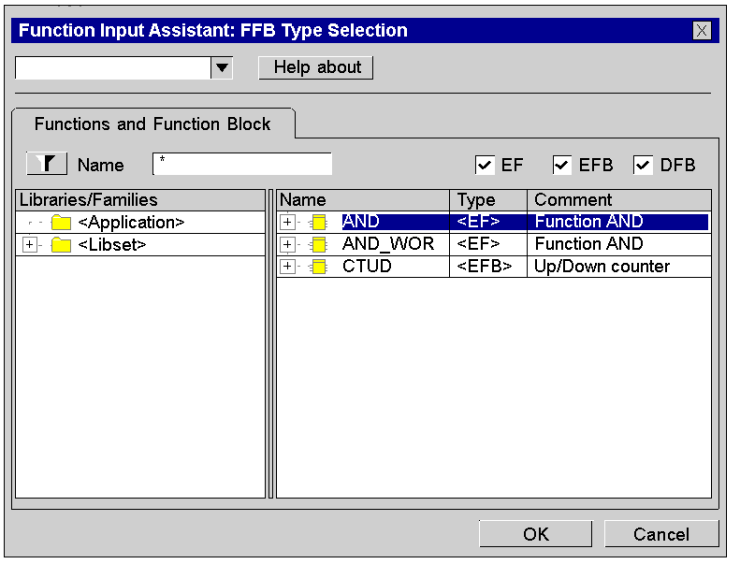
**NOTE:** Procedures are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools** → **Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow procedures** check box.

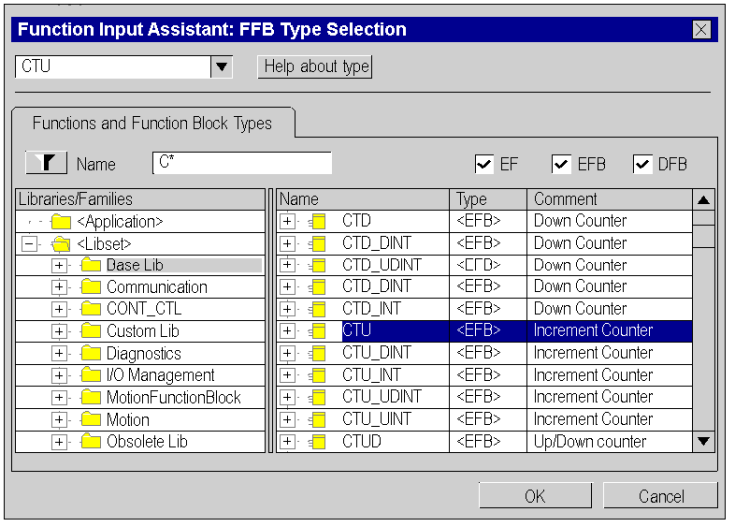
## Calling a function block using the function input assistant

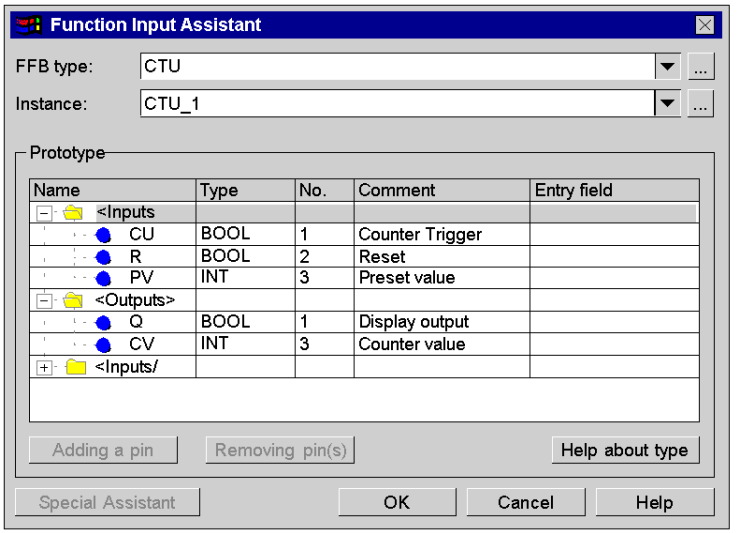
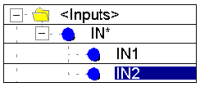
Process the following steps to call a function block (elementary or derived) with the function input assistant:

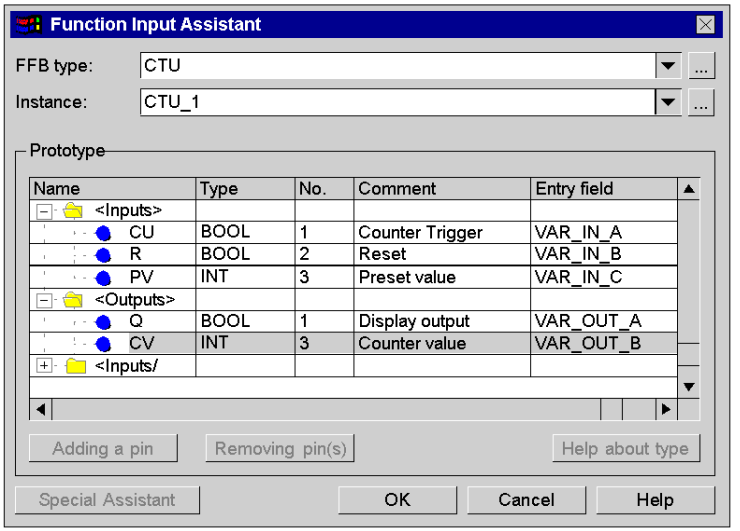
| Step | Action |
|---|---|
| 1 | Place the cursor at the target position.<br>Note: No text may be selected in the section.<br>The following positions are possible:<br>● On a free position outside of an existing FFB call.<br>● On a space within an existing FFB call. (In this case, an FFB call is created within an FFB call.) |

| Step | Action |
|---|---|
| 2 | Open the function input assistant *(see page 1194)* using:<br>● the **Edit → FFB Input Assistant... menu command.**.<br>● The **FFB Input Assistant...menu command**from the shortcut menu (right-click) for the section.<br>● Pressing the **Ctrl+I** key combination.<br>or<br>● Select the  symbol.<br>**Result:**<br>The function input assistant will be opened.<br>  |

| Step | Action |
|---|---|
| 3 | In the **FFB type** line click the button **....**, to open an FFB selection window *(see page 1192)*.<br>**Result:**<br>The FFB selection dialog box is opened.<br>  |
| 4 | Select the library and family from the **Libraries/Families** column which contains the desired function block.<br>**Tip:** If you do not know where to find the FFB you are looking for select the **Libset**, entry to see a list of all the FFBs available independently of their libraries. |

| Step | Action |
|---|---|
| 5 | Select the desired function block from the **Name** column.<br>Example:<br>  |

| Step | Action |
|---|---|
| 6 | Confirm the selection with **OK**.<br>**Result:**<br>● The function block is added to the function input assistant dialog box.<br>● An instance name is automatically suggested. You can keep this or change it as you please.<br>● You can see all available parameters *(see Unity Pro, Program Languages and Structure, Reference Manual )* of the function block, see also Information about programming *(see Unity Pro, Program Languages and Structure, Reference Manual )*.<br>  |
| 7 | The number of inputs can be increased with some elementary functions. (You find out which functions can be expanded by referring to the descriptions for the individual functions.)<br>Expandable pins can be recognized from the following display:<br>  <br>To add further pins select the last pin in the structure and press **Add pin**. |

| Step | Action |
|---|---|
| 8 | Double-click the **Entry field** cell of the first parameter and enter the name of the variable/address or an expression to be used.<br>You have the following options to enter the variable/address:<br>● You can now enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the button **...** to open a variable selection dialog box *(see page 1189)*.<br>Assign all parameters of the function block a variable/address in this way.<br>Example:<br>  |
| 9 | Confirm the entries using the **OK** button.<br>**Result:** The function block call is inserted into the IL section and a syntax and semantics check *(see page 1045)* is performed.<br>Example:<br>`CAL CTU_1 (`<br>`CU := VAR_IN_A,`<br>`R := VAR_IN_B,`<br>`PV := VAR_IN_C,`<br>`CV => VAR_OUT_B,`<br>`Q => VAR_OUT_A`<br>`)` |

## Calling a function or procedure using the function input assistant

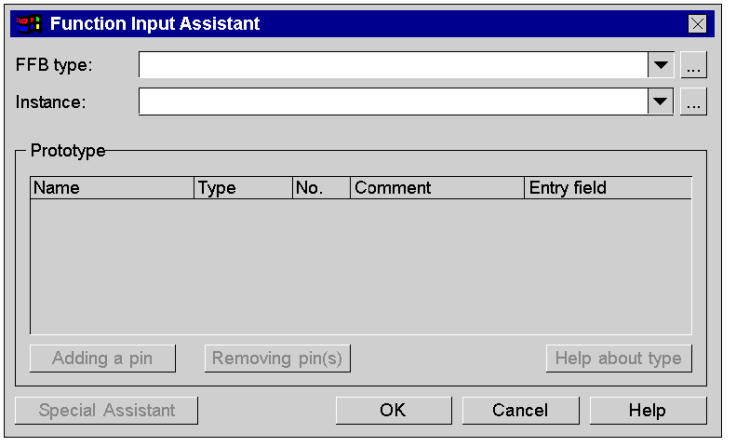
Calling a function or procedure using the function input assistant is identical in principle to calling a function block. The only difference is the absence of an instance name (step 6 of the procedure *Calling a function block using the function input assistant, page 1057*).

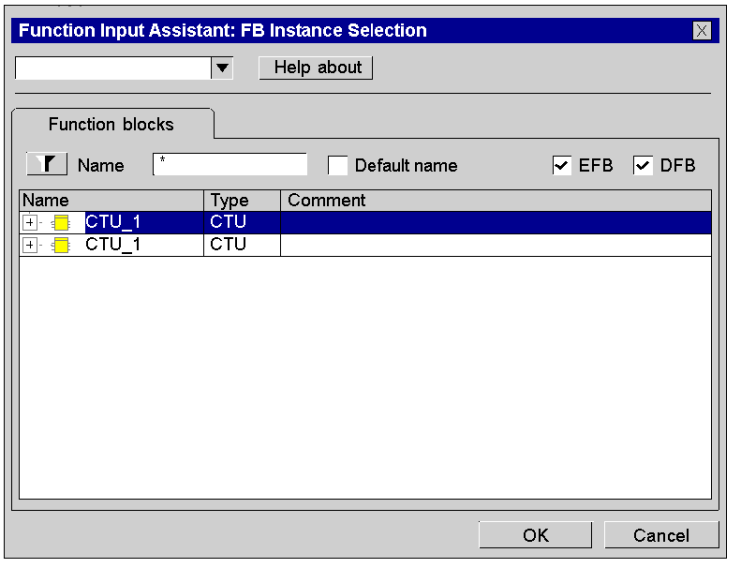
## Remarks

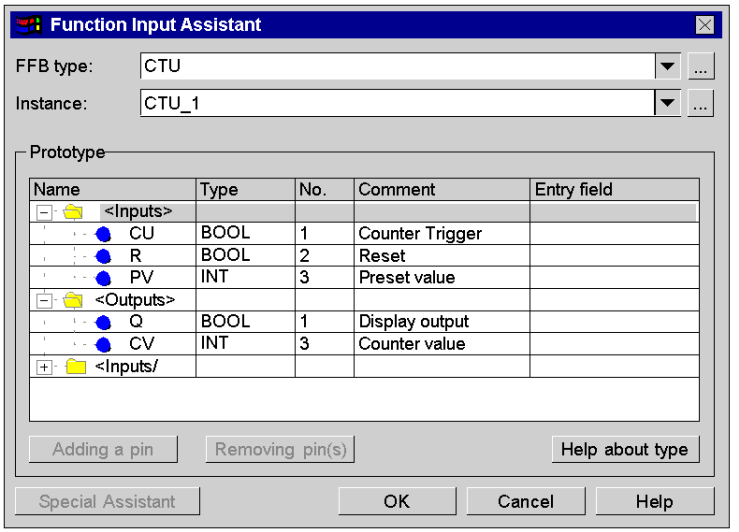
Some complex FFBs provide additional windows for entering parameters. Access to this window is gained using the **Special Assistant** button that appears at the bottom of the window.

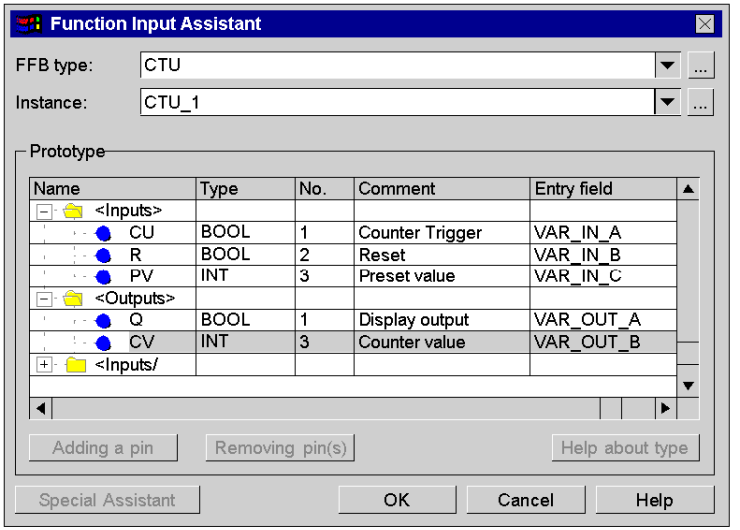
## Calling a function block instance using the function input assistant

Process the following steps to select a block instance that has already been declared:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | Open the function input assistant *(see page 1194)* using:<br>● The **Edit → FFB Input Assistant** menu command.<br>● The **FFB Input Assistant...menu command**from the shortcut menu (right-click) for the section.<br>● Pressing the **Ctrl+I** key combination.<br>or<br>● Select the  symbol.<br>**Result:**<br>The function input assistant will be opened.<br>  |

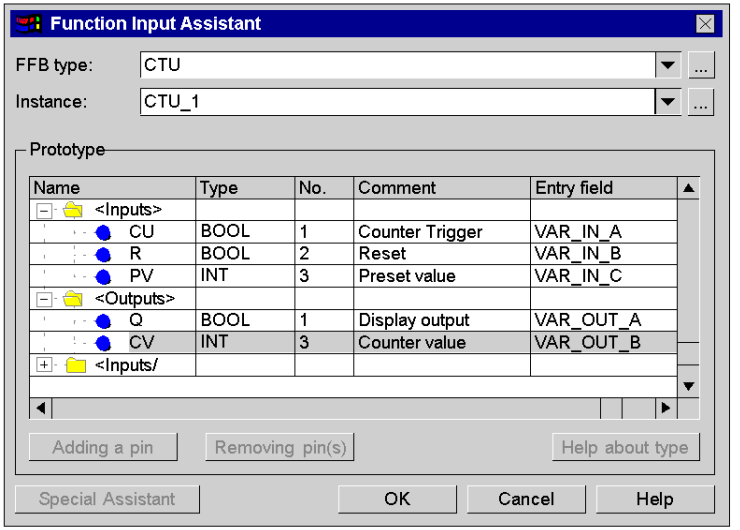
| Step | Action |
|---|---|
| 3 | In the **Instance line** click the button **....**, to open an FFB selection window *(see page 1184)*.<br>**Result:**<br>The dialog box for selecting function block instances is opened.<br>  <br> |

| Step | Action |
|---|---|
| 4 | Select the desired block instance from the **Name** column and confirm the selection with **OK**.<br>**Result:**<br>● The block instance is added to the function input assistant dialog box.<br>● You can see all available parameters *(see Unity Pro, Program Languages and Structure, Reference Manual )* of the function block, see also Information about programming *(see Unity Pro, Program Languages and Structure, Reference Manual )*.<br>  |

| Step | Action |
|---|---|
| 5 | Double-click the **Entry field** cell of the first parameter and enter the name of the variable/address to be used.<br>You have the following options to enter the variable/address:<br>● You can now enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the button **...** to open a variable selection dialog box *(see page 1189)*.<br>Assign all parameters of the function block a variable/address in this way.<br>Example:<br>  |
| 6 | Confirm the entries using the **OK** button.<br>**Result:** The function block call is inserted into the IL section and a syntax and semantics check *(see page 1045)* is performed. If the variables are not declared yet a red wavy line under the name is used to identify them.<br>Example:<br>`CAL CTU_1 (`<br>`CU := VAR_IN_A,`<br>`R := VAR_IN_B,`<br>`PV := VAR_IN_C,`<br>`CV => VAR_OUT_B,`<br>`Q => VAR_OUT_A`<br>`)` |
| 7 | Declare all variables in the Data Editor *(see page 370)* or directly in the IL section *(see page 1053)*.<br>**Result:** The red wavy line under the variable name disappears. |

## Changing the actual parameter using the function input assistant

Process the following steps to change actual parameters with the function input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor inside a word for the FFB to be changed.<br>Note: No text may be selected and the FFB call to be changed must be syntactically correct. |
| 2 | Open the function input assistant *(see page 1194)* using:<br>● The **Edit → FFB Input Assistant... menu command.**.<br>● The **FFB Input Assistant...menu command**from the shortcut menu (right-click) for the section.<br>● Pressing the **Ctrl+I** key combination.<br>or<br>● Select the  symbol.<br>**Result:**<br>The function input assistant is opened with the current actual parameters.<br>  |
| 3 | Make the desired changes and confirm using **OK**. |



## Inserting FFBs via drag & drop

FFBs can also be inserted into the section via drag & drop using the **Types Library Browser**.

You have the following options to activate the **Types Library Manager**:

- Use the **Tools → Types Library Manager** menu command.
  or
- Pressing the **Alt+3** key combination.

# Using Public Variables

## Introduction

In addition to inputs/outputs, some function blocks also provide public variables.

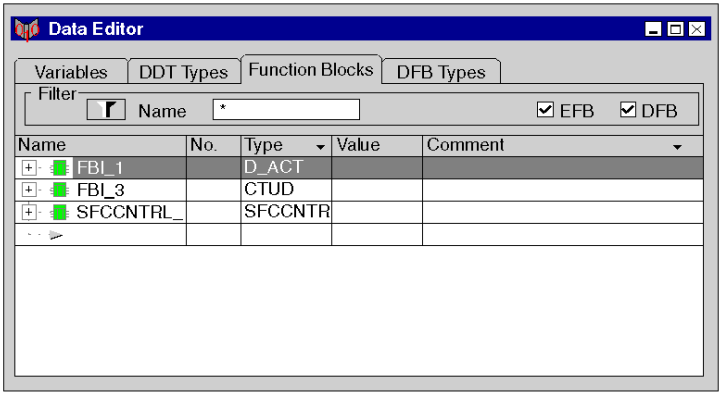
These variables transfer statistical values (values that are not influenced by the process) to the function block. They are used for setting parameters for the function block.

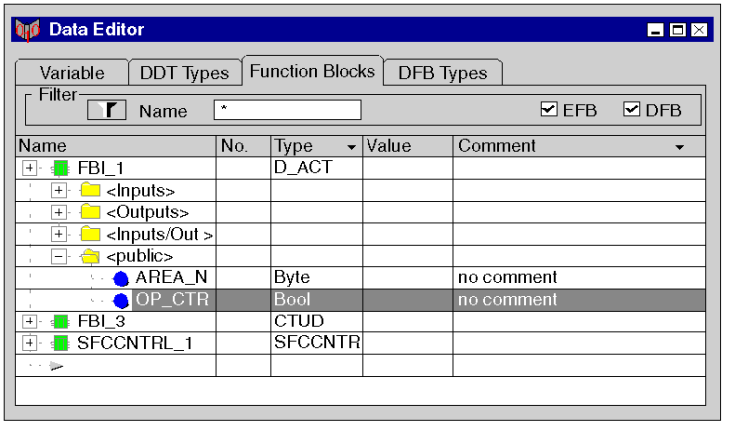
The assignment of values to public variables is made via their initial values or via the load and save instructions.

Public variables are read via the instance name of the function block and the names of the public variables.

## Assigning Values using the Initial Value

Process the following steps to assign values using the initial value:

| Step | Action |
|---|---|
| 1 | Call the function block for whose public variable you want to assign a value, see also *Calling an FFB, page 1057*. |
| 2 | Open the data editor, see also *Data Editor Access, page 328*. |
| 3 | Select the **Function Block** tab.<br>**Result:**<br>The function block instances are displayed.<br>  |

| Step | Action |
|---|---|
| 4 | Expand the display of the public variables by clicking the respective **+** symbols.<br>**Result:**<br>You can now see all available public variables of the function block.<br>  |
| 5 | Enter the desired value for the public variable in the **Value** box. |

## Assigning Values using the Load and Save Instructions

Carry out the following steps to assign values using the load and save instructions:

| Step | Action |
|---|---|
| 1 | Call the function block for whose public variable you want to assign a value, see also *Calling an FFB, page 1057*. |
| 2 | Load the value to be assigned to the buffer.<br>For example:<br>`LD 1` |
| 3 | Save the buffer contents in a public variable, see also *Using formal parameters, page 1054*.<br>For example:<br>`ST D_ACT1.OP_CTRL` |

## Reading public variables

See *Using formal parameters, page 1054*

# Online functions

## Online functions

The description of online functions is found in chapter *Debugging in Textual Languages (Structured Text, Instruction List)*, *page 1431*.

# Reference Data Type in IL

## Introduction

References to variables can be assigned in IL application programs.

A reference can be de-referenced in IL application programs.

For more information, refer to Reference Data Type *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

IL Examples using the Reference Data Type:

myIL : [MAST]

```
REF (IN := myVarInt(*ANY*))
ST ptrToMyInt(*REF_ANY*)

MOVE (IN := REF(myVarBool)(*ANY*))
ST ptrToMyBool(*ANY*)

CAL TON_3  (IN := ptrToMyBool(*BOOL*),
            PT := t#5s(*TIME*),
            Q => output(*BOOL),
            ET => internalTime(*TIME*))
```

The EF, `REF`, takes a variable as an input parameter (type `ANY`) and provides its address as output parameter (type `REF_ANY`).

The EF, `MOVE`, can assign a reference to a variable to another reference. If the input parameter is (`REF(MyVariable)`), `MOVE` has the same effect as the `REF` EF.

The EF, `TON`, takes `ptrToMyBool^` as an input parameter to dereference the reference ptrToMyBool.

# Export/Import IL Sections

## Export/Import

The description for exporting/importing sections is found in the chapter *Import / Export*, *page 1719*.

# Chapter 26
# ST Editor

## Overview

This chapter describes the specific menus and dialog boxes of the ST editor.

For a description of the syntax for the ST programming language, please refer to the Structured Text ST *(see Unity Pro, Program Languages and Structure, Reference Manual )* chapter in the Reference manual.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Structure of an ST program | 1074 |
| Creating an ST Program | 1076 |
| Syntax and Semantics Check during Programming | 1077 |
| Navigating with the keyboard | 1081 |
| Data input | 1082 |
| Calling an FFB | 1088 |
| Using public variables | 1099 |
| Calling a statement | 1101 |
| Online functions | 1107 |
| Reference Data Type in ST | 1108 |
| Export/Import IL Sections | 1109 |

# Structure of an ST program

## Introduction

An ST program (Structured Text) consists of a list of statements that are executed in sequence by the controller. Using these statements you can call function blocks, functions and procedures conditionally or unconditionally, make assignments, execute statements explicitly, repeat statements and execute jumps conditionally or unconditionally within a section.

## Properties of an ST program

Properties of an ST program:

- Statements consist of:
  - an operator,
  - an operand,
  - an expression if required and
  - a comment if required.
  - In addition, every statement can be provided with a label.
- Statements must be concluded with semicolons.
- Several statements (separated by semicolons) may be present in one line.
- A single semi-colon denotes an empty statements *(see Unity Pro, Program Languages and Structure, Reference Manual )*.
- Each line is limited to 300 characters.
- Line breaks are possible in statements (multi-line statements).
- Labels, symbols and comments can be placed anywhere in a section. (Comments can be entered anywhere where empty spaces are permitted.)
- A syntax and semantics check *(see page 1077)* is performed immediately after the statement is entered. The result of this check is displayed in colored text.
- Syntactically or semantically incorrect sections can be saved.

## Edit and view functions

Edit and view functions of the ST editor:

- Entering text using the insert or overwrite mode *(see page 1076)*
- Selecting text *(see page 1008)* (characters, words, lines, all)
- Deleting text *(see page 1010)*
- Cutting *(see page 1010)*, copying *(see page 1010)* and pasting *(see page 1011)*
- Moving text *(see page 1011)* (also between different ST/IL sections)
- Undo *(see page 1012)* and Redo *(see page 1012)*
- Go To function *(see page 1017)*
- Using bookmarks *(see page 1015)*
- Searching for text and marking the result *(see page 1022)*
- Searching and replacing text *(see page 1024)*
- Searching and replacing variables and function blocks and DFB instances
- Displaying text in selectable fonts and sizes
  The text display type can be selected from the **View** → **Select Font** menu command. This configuration applies to all sections of the IL and ST programming languages.
- Intelligent text indentation
  The text indentation of a new line is taken from the previous one.
- Display the position (line, column) of the cursor in the status bar.

## Input assistants

Input assistant for the ST editor:

- Input assistant for variables *(see page 1082)*
- Input assistant for functions, function blocks and procedures *(see page 1088)*
- Input assistant for statements *(see page 1101)*

## Online functions

ST editor online functions:

- Displaying actual values *(see page 1030)*.
- Setting breakpoints *(see page 1432)*
- Setting watchpoints *(see page 1437)*
- Step by Step *(see page 1434)*

# Creating an ST Program

## Writing in insert/overwrite mode

Text can be entered in insert or overwrite modes

- Insert mode
  The characters entered are inserted in the current position of the insert marker in addition to the existing characters.
- Overwrite mode
  The characters entered are inserted in the current position of the insert marker and overwrite the existing characters.

Switching between the modes is done using the **Insert** key.

The current mode is displayed by **INS** for insert mode and **OVR** for overwrite mode.

## Creating an ST program

Carry out the following procedures to create an ST program:

| Step | Action |
|---|---|
| 1 | Creating an ST section *(see page 458)*. |
| 2 | Enter the first instruction (e.g. `VarA := VarB`).<br>**Note:** As soon as the text is entered several checks are made, such as syntax/semantic error checks, correct spelling of keywords and variable names etc. A detailed description of syntax is found in chapter Structured Text *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual.<br>The results of the checks are indicated by a color folder, see also *Syntax and Semantics Check during Programming, page 1077*. |
| 3 | Terminate the instruction with the closing operator (`;`). |
| 4 | Confirm the line with the **ENTER** key. |
| 5 | Repeat these steps until all instructions are entered.<br>Example:<br>`VarA := VarB*VarB-4*VarC*VarD ;`<br>`IF VarA < 0.0 THEN VarE := 0 ;`<br>`ELSIF VarA = 0.0 THEN`<br>`VarE := 1 ;`<br>`X1 := -X2/(2.0*VarC) ;`<br>`ELSE`<br>`VarE := 2;`<br>`X1 := (-B+SQRT(D)) / (2.0*A) ;`<br>`X2 := (-B-SQRT(D)) / (2.0*A) ;`<br>`END_IF;` |

# Syntax and Semantics Check during Programming

## Introduction

Syntax and semantics check is performed directly when creating the program.

The result of this check is displayed in three formats:

- directly in the program section with colored text
- as tooltip if the cursor is placed on incorrect text.
- In the output window, if **Generate** → **Analyze** is selected.

## Illustration in the program section

Representation of colors and labels:

| Labeling | Description | Example |
|---|---|---|
| blue | Keyword (e.g. `IF`, `FOR`, `WHILE`,...)<br>**Note:** IL specific keywords (e.g. `LD`, `CAL`, `ST`) are not keywords in ST, but are still labeled as such for technical reasons. | `IF VarA` |
| red | Operators (that are not keywords) | `My_Count (CU:=Var1,R:=Var2, PV:=Var3,Q=>Var4);` |
| green | Comment | `IF VarA (* Comment *)` |
| black | standard text (e.g. variable, address, label, ...) | `start: IF VarB` |
| bold | The name of DFBs and subroutines are shown in boldface to identify that these objects can be refined *(see page 1028)*. | **`My_DFB`** `(IN1:=Var1, IN2:=Var2, OUT=>Var3);`<br>**`MY_SUBROUTINE`** `();` |
| underlining | In DFB sections, the formal parameters of the DFBs are shown underlined. | `OUT := IN1 AND IN2;` |

| Labeling | Description | Example |
|---|---|---|
| red wavy line | Faulty text:<br>● Syntax error<br>e.g. spelling errors in keywords, non-declared variables or FB instances, incorrect block call, unvailable block call (in the current PLC plateform or in the library types), EFB type call instead of the instance,...<br>● Semantic error<br>e.g. division by zero, value range exceeded/underflow (recognized by the system when entered) | `IFs i1`<br>`TON();` |
| Inverted text | Text is selected | `IF VarA` |

## Tooltips

If the cursor is placed over incorrect text, the tooltip is displayed with a brief description of the cause of the error. It will also be displayed in the output window after the analysis.

## Error messages in the output window

Error message: `Accessed object is not a function.`

| Cause of error | Error Correction | Example |
|---|---|---|
| Write error is present during accessing of a function. | Correct the write error. | Wrong:<br>`VarG := LEQ (VarE, VarF);`<br>Right:<br>`VarG := LE (VarE, VarF);` |
| Name of the function was already used for:<br>● one variable<br>● an FB instance<br>● a DFB Type<br>● an SR Section<br>● a procedure or function in a user library | Rename the already used object. | - |

| Cause of error | Error Correction | Example |
|---|---|---|
| Wrong syntax used during access<br>● an FB<br>● a procedure<br>● an SR Section | Correct the syntax. | Wrong:<br>`VarK := My_CTD (VarH, VarI, VarJ, VarL);`<br>Right:<br>`My_CTD (VarH, VarI, VarJ, VarK, VarL);`<br>Wrong:<br>`VarQ := DEC (VarQ);`<br>Right:<br>`DEC (VarQ);`<br>Wrong:<br>`VarA := My_SR ();`<br>Right:<br>`My_SR ();` |

Error message: `Accessed object is not a function block.`

| Cause of error | Error Correction | Example |
|---|---|---|
| Write error is present during accessing.<br>● an FB<br>● a procedure<br>● an SR Section | Correct the write error. | Wrong:<br>`MyCTD (VarH, VarI, VarJ, VarK, VarL);`<br>Right:<br>`My_CTD (VarH, VarI, VarJ, VarK, VarL);`<br>Wrong:<br>`DEZ (VarQ);`<br>Right:<br>`DEC (VarQ);`<br>Wrong:<br>`MySR ();`<br>Right:<br>`My_SR ();` |
| Name of the function was already used for:<br>● one variable<br>● an FB instance<br>● a DFB Type<br>● an SR Section<br>● a procedure or function in a user library | Rename the already used object. | - |
| Function block instances that are not defined in the data editor. | Define the FB on the data editor. | - |

| Cause of error | Error Correction | Example |
|---|---|---|
| EFB that is not available for the current PLC platform. | Change the EFB. | - |
| An EFB that is not in the libset of the current project. | Update the libset. | - |
| Wrong syntax used during access.<br>● a function<br><br><br>● a function block | Correct the syntax. | Wrong:<br>`DIVTIME (VarR, VarS);`<br>`DIVTIME (VarR, VarS, VarT);`<br>Right:<br>`VarT := DIVTIME (VarR, VarS);`<br>Wrong:<br>`CTD (VarH, VarI, VarJ, VarK, VarL);`<br>Right:<br>`My_CTD (VarH, VarI, VarJ, VarK, VarL);`<br>Wrong:<br>`TON ();`<br>Right:<br>`My_TON ();` |

# Navigating with the keyboard

## Navigating with the keyboard

The following keys and key combinations are provided for navigation:

| Key combinations | Motion |
|---|---|
| **Cursor left** | Moves the cursor to the left |
| **Cursor right** | Moves the cursor to the right |
| **Cursor up** | Moves the cursor up |
| **Cursor down** | Moves the cursor down |
| **Ctrl**+**Cursor left** | Moves the cursor one string left |
| **Ctrl**+**Cursor right** | Moves the cursor one string right |
| **Home** | Moves the cursor to the beginning of the current line |
| **End** | Moves the cursor to the end of the current line |
| **Ctrl**+**Home** | Moves the cursor to the beginning of the current section |
| **Ctrl**+**End** | Moves the cursor to the end of the current section |
| **Page Up** | Scrolls one page up (the cursor remains in its position on the screen) |
| **Page Down** | Scrolls one page down (the cursor remains in its position on the screen) |
| **Ctrl**+**Alt**+**Page Up** | Displays the previous section (including via the **View** → **Previous Section**menu). |
| **Ctrl**+**Alt**+**Page Down** | Displays the next section (including via the **View** → **Next Section**menu). |
| **Alt**+**Enter** | Opens the properties dialog box for the selected section. |

# Data input

## Introduction

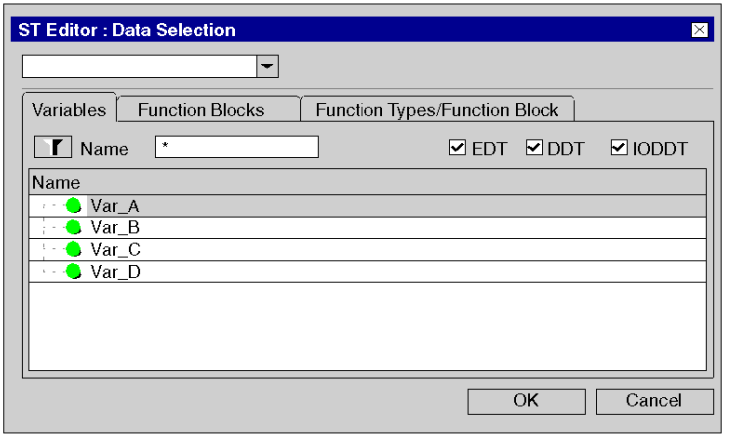
You have different options for using data in your ST section.

- Using already declared variables *(see page 1082)*
  - ❍ You can use a data selection dialog box.
    or
  - ❍ you can enter the variable name just like any other text.
- Use undeclared variables and declare them later *(see page 1083)*
  You can define the variable name when creating your ST section and then declare all variables used.
- Use undeclared variables and declare them immediately *(see page 1084)*
  You can define variable names when creating your ST section and declare them immediately.
- Declaration of variables directly in the ST section. *(see page 1084)*
  You can declare variables in the Data Editor *(see page 370)*, but also directly in the ST section.
- Using formal parameters from function blocks (inputs/outputs, public variables) *(see page 1085)*
  - ❍ You can use a data selection dialog box.
    or
  - ❍ You can enter the formal parameter name just like any other text.

## Entering declared variables using data selection

Carry out the following steps to enter a declared variable using data selection:

| Step | Action |
|---|---|
| 1 | Declare the required variables, see also *Creation of EDT type variable instances, page 370*. |
| 2 | Place the cursor at the target position. |
| 3 | Open data selection by:<br>● The **Edit → Data Selection menu command.**.<br>● The **Data Selection... menu command**from the shortcut menu for the section.<br>or<br>● Press the **Ctrl**+**D** key combination.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |

| Step | Action |
|---|---|
| 4 | You have the following options to enter variables:<br>● You can now enter the variable name and confirm using **Enter** or the button ✓.<br>● You can select the variable names from the list of recently used names using the ▼ symbol and confirm using **Enter** or the button ✓.<br>● You can delete your entry by pressing **Esc** or the button ✗.<br>● Use the button **...**to open a variable selection dialog box *(see page 1189)* and in the **Variables** tab confirm the selected variable using the **OK** button.<br>Example of a variable selection dialog box:<br>  <br><br>**Result:** The selected variable is enabled in the section. |

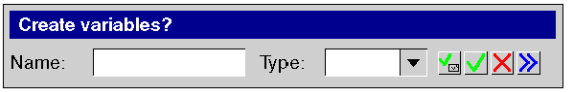
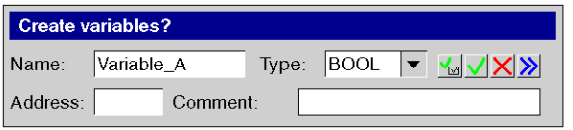
## Entering undeclared variables and declaring them later

Process the following steps to enter undeclared variables and declare them later:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | Enter the variable name.<br>**Result:** The variable selected is accepted. The syntax and semantics check *(see page 1077)* marks the variable name with a red wavy line to identify it as being undeclared, e.g.<br>VarA |
| 3 | Enter all variables in this way. |
| 4 | Declare all variables in the Data Editor *(see page 370)* or directly in the ST section *(see page 1084)*.<br>**Result:** The red wavy line under the variable name disappears. |

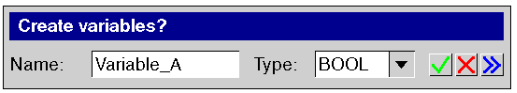
## Entering undeclared variables and declaring them immediately

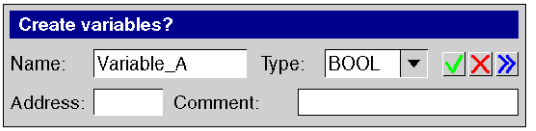
Process the following steps to enter undeclared variables and declare them immediately:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | Click **Create Variable** in the shortcut menu, or press the **Shift**+**Enter** key combination<br>**Result:** The variable declaration dialog box opens.<br>  |
| 3 | Enter the variable name and data type. |
| 4 | If you do not want to assign an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and inserted at the target position in the section. |
| 5 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 6 | Confirm with **Enter** or the  symbol.<br>**Result:** The variable is declared and inserted at the target position in the section. |

## Declaration of variables directly in the ST section.

Process the following steps to declare variables directly in the ST section.

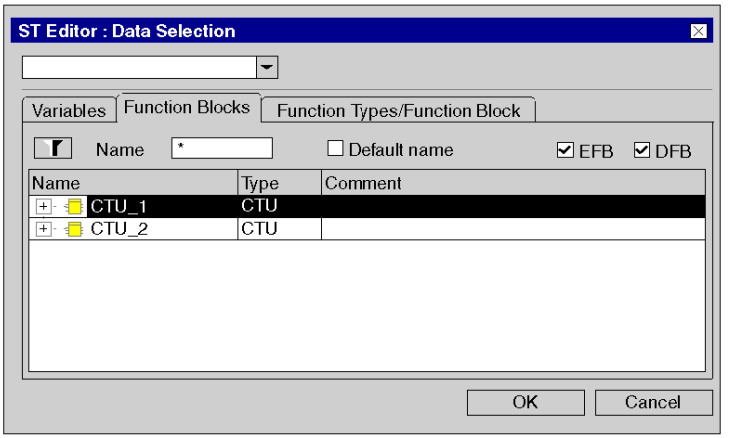
| Step | Action |
|---|---|
| 1 | Place the cursor in an undeclared variable (undeclared variables are identified by a red wavy line). |
| 2 | Click **Create Variable** in the shortcut menu, or press the **Shift**+**Enter** key combination<br>**Result:** The variable declaration dialog box opens.<br>  <br>**Note:** The variable name automatically appears in the **Name** field, if editor recognizes the text entered in the section as a variable and its syntax is correct. |

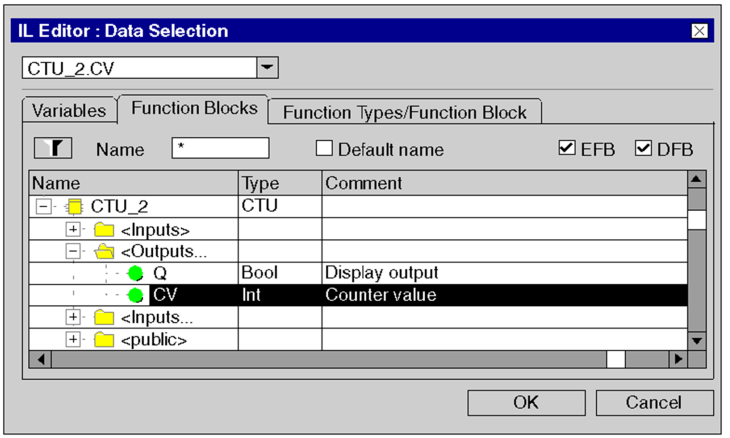
| Step | Action |
|---|---|
| 3 | If you do not want to assign an address or comment, confirm using **Enter** or the  symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears. |
| 4 | If you do want to assign the variable an address and/or comment, use the  symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 5 | Confirm with **Enter** or the  symbol.<br>**Result:** The variable is declared and the red wavy line under the variable name disappears. |

## Using formal parameters

Process the following steps to use formal parameters with data selection:

| Step | Action |
|---|---|
| 1 | Call the function block whose formal parameters you want to use, see also Calling a function block *(see page 1089)*. |
| 2 | To use the formal parameter, place the cursor at the target position. |
| 3 | Open data selection by:<br>● The **Edit** → **Data Selection menu command.**.<br>● The **Data Selection... menu command**from the shortcut menu for the section.<br>or<br>● Press the **Ctrl**+**D** key combination.<br>**Result:**<br>A data selection box *(see page 1911)* is opened.<br>  |

| Step | Action |
|---|---|
| 4 | You have the following options to enter the formal parameter:<br>● You can enter the formal parameter name and confirm using **Enter** or the ✓ button.<br>In this case the procedure ends here.<br>● You can select the name from the list of recently used names using the ▼ symbol and confirm using **Enter** or the button ✓.<br>In this case the procedure ends here.<br>or<br>● Using the button **...** to open a data selection dialog box *(see page 1189)*. |
| 5 | Select the **Function Blocks** tab.<br>**Result:**<br>All declared block instances are displayed.<br>  |

| Step | Action |
|---|---|
| 6 | Expand the display of the inputs, outputs, and any inputs/outputs and public variables by clicking their **+** symbols.<br>**Result:**<br>All available formal parameters are displayed now.<br>  |
| 7 | Select the desired formal parameter and confirm with **OK**.<br>**Result:** The selected formal parameter is inserted into the section and a syntax and semantics check *(see page 1077)* is performed.<br>For example:<br>`IF CTU_2.CV = 35 THEN` |

# Calling an FFB

## Introduction

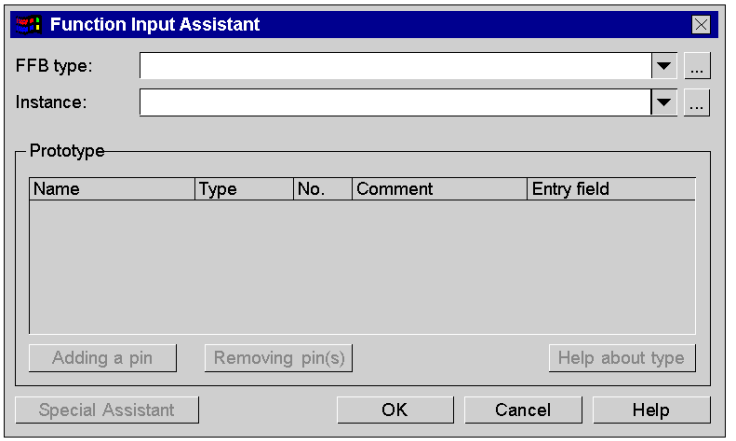
You have the following options to open FFBs (Elementary Function, Elementary Function Block, Derived Function Block, Procedure):

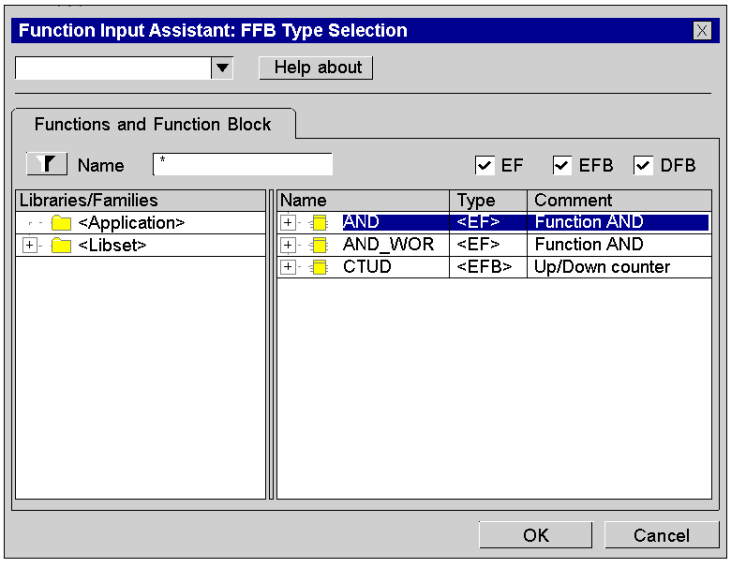
- You can enter the call like you enter any text (for syntax, see the section Calling FFBs *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual),
- You can use the function input assistants,
  or
- You can enter the FFB to the section using the drag & drop method.

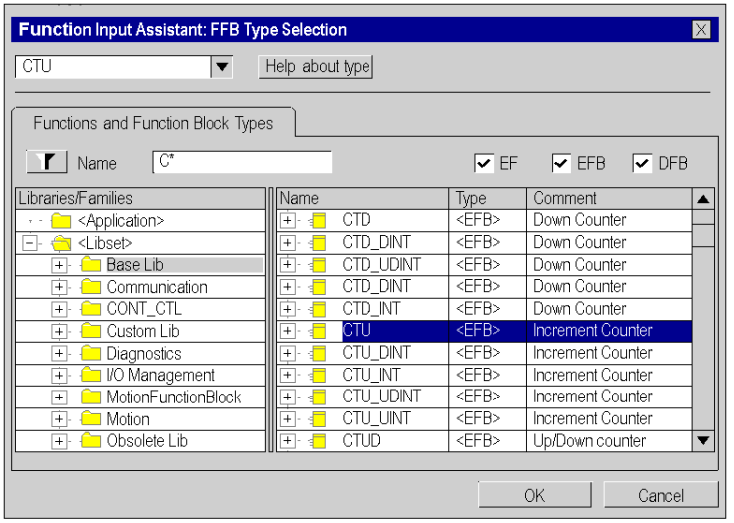
**NOTE:** Procedures are expansions to IEC 61131-3 and must be explicitly enabled via the **Tools → Project Settings** dialog box, in the **Language extensions** tab by activating the **Allow procedures** check box.

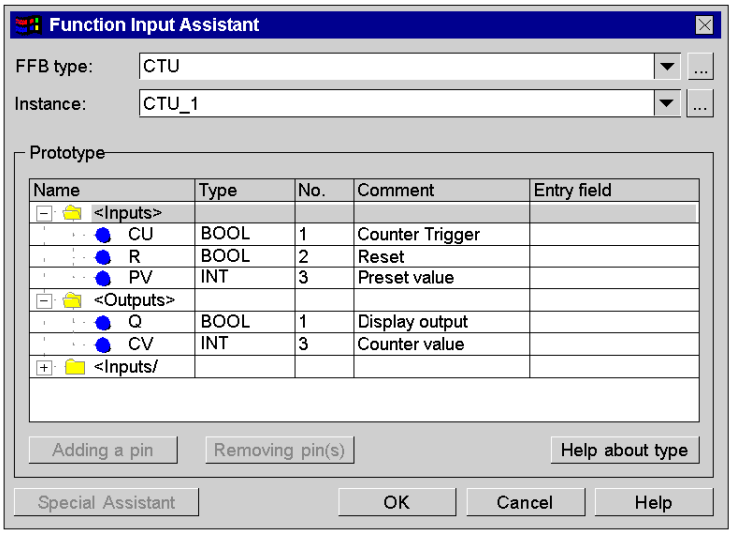
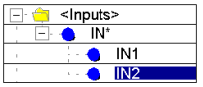
## Calling a function block using the function input assistant

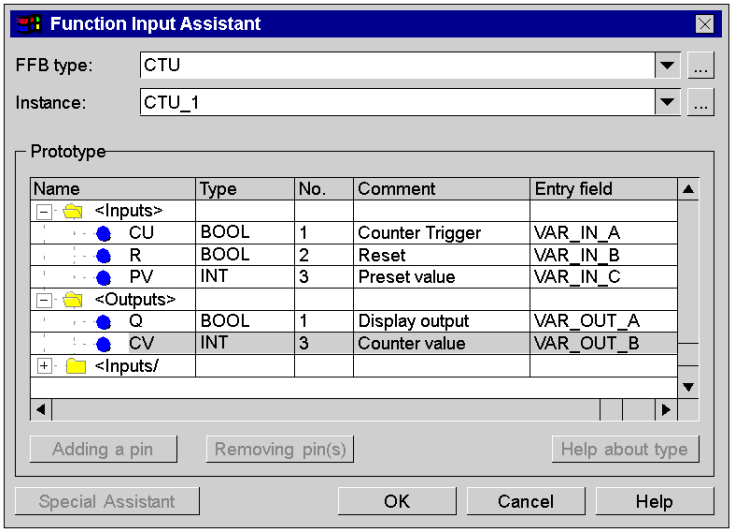
Process the following steps to call a function block (elementary or derived) with the function input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position.<br>Note: No text may be selected in the section.<br>The following positions are possible:<br>● On a free position outside of an existing FFB call.<br>● On a space within an existing FFB call. (In this case, an FFB call is created within an FFB call.) |
| 2 | Open the function input assistant *(see page 1194)*:<br>● The **Edit → FFB Input Assistant... menu command.**.<br>● The **FFB Input Assistant...menu command**from the shortcut menu (right-click) for the section.<br>● Press the **Ctrl+I**key combination.<br>or<br>● Select the  symbol.<br>**Result:**<br>The function input assistant will be opened.<br>  |

| Step | Action |
|---|---|
| 3 | Click the **FFB-Typ** button in the**....column**, to open an FFB selection window *(see page 1192)*.<br>**Result:**<br>The FFB selection dialog box is opened.<br>  |
| 4 | Select the library and family from the **Libraries/Families** column which contains the desired function block.<br>**Tip:** If you do not know where to find the FFB you are looking for select the **Libset**, entry to see a list of all the FFBs available independently of their libraries. |

| Step | Action |
|---|---|
| 5 | Select the desired function block from the **Name** column.<br>Example:<br>  |

| Step | Action |
|---|---|
| 6 | Confirm the selection with **OK**.<br>**Result:**<br>● The function block is added to the function input assistant dialog box.<br>● An instance name is automatically suggested. You can keep this or change it as you please.<br>● You can see all available parameters *(see Unity Pro, Program Languages and Structure, Reference Manual )* of the function block, see also Information about programming *(see Unity Pro, Program Languages and Structure, Reference Manual )*.<br>  |
| 7 | The number of inputs can be increased with some elementary functions. (You find out which functions can be expanded by referring to the descriptions for the individual functions.)<br>Expandable pins can be recognized from the following display:<br>  <br>To add further pins select the last pin in the structure and press **Add pin**. |

| Step | Action |
|---|---|
| 8 | Double-click the **Entry field** cell of the first parameter and enter the name of the variable/address or an expression to be used.<br>You have the following options to enter the variable/address:<br>● You can now enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the button **...** to open a variable selection dialog box *(see page 1189)*.<br>Assign all parameters of the function block a variable/address in this way.<br>Example:<br>  |
| 9 | Confirm the entries using the **OK** button.<br>**Result:** The function block call is inserted into the ST section and a syntax and semantics check *(see page 1077)* is performed.<br>Example:<br>`CTU_1 (CU := VAR_IN_A,`<br>`R := VAR_IN_B,`<br>`PV := VAR_IN_C,`<br>`CV => VAR_OUT_B,`<br>`Q => VAR_OUT_A);` |

## Calling a function or procedure using the function input assistant

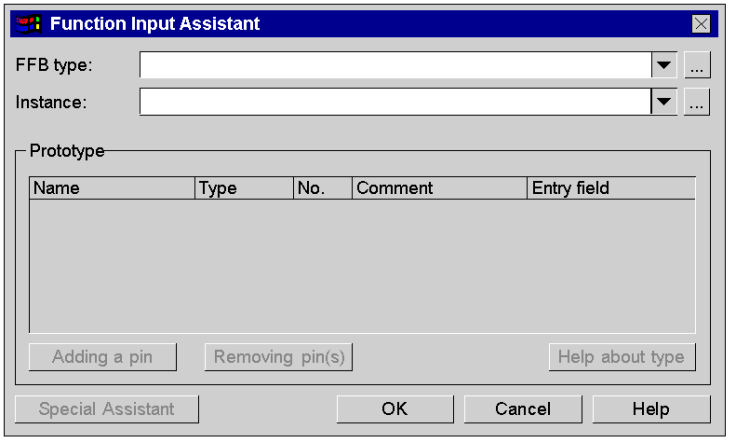
Calling a function or procedure using the function input assistant is identical in principle to calling a function block. The only difference is the absence of an instance name (step 5 of the procedure *Calling a function block using the function input assistant, page 1089*).

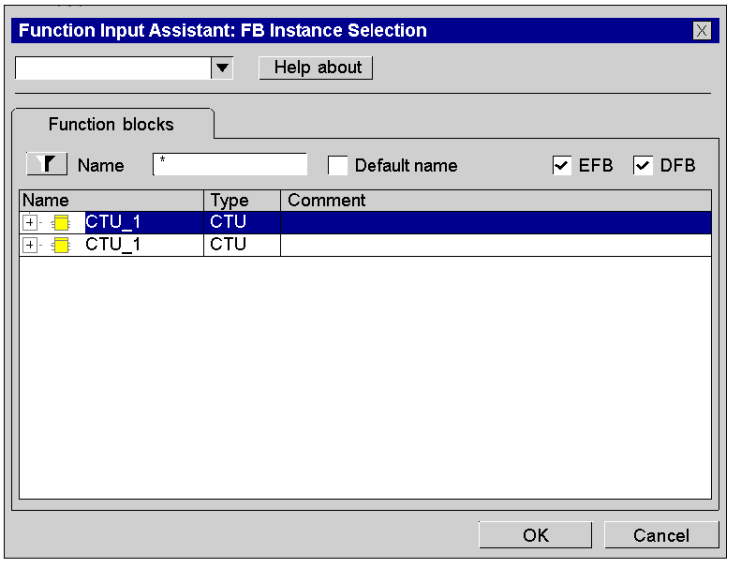
## Remarks

Some complex FFBs provide additional windows for entering parameters. Access to this window is gained using the **Special Assistant** button that appears at the bottom of the window.

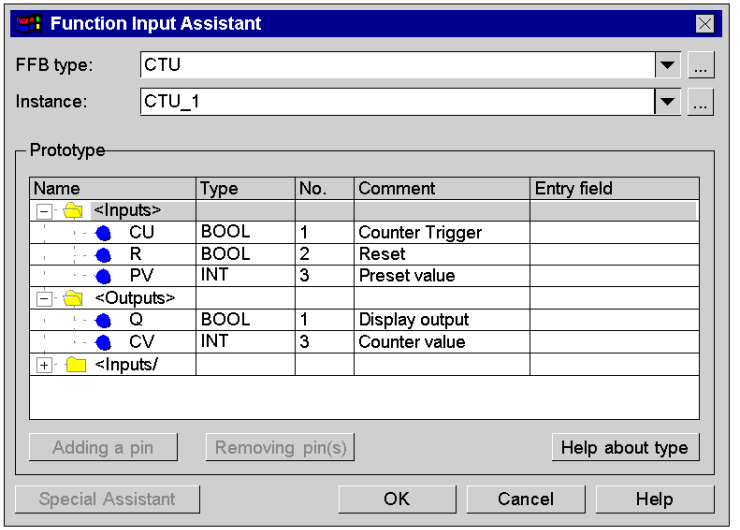
## Calling a function block instance using the function input assistant

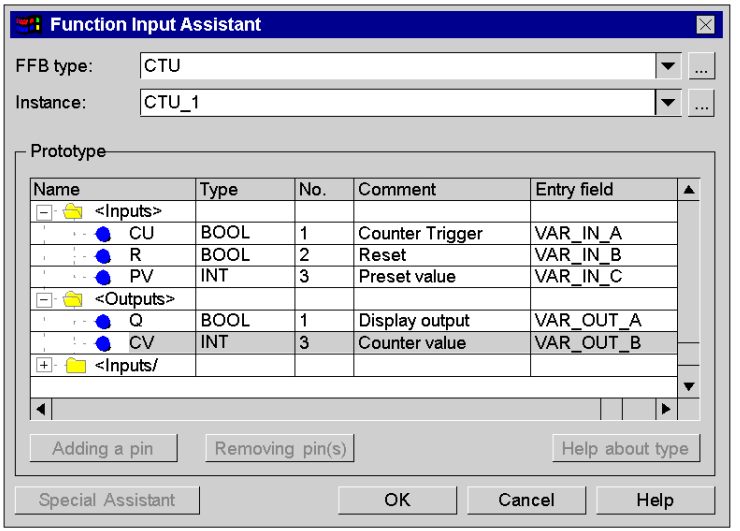
Process the following steps to select a block instance that has already been declared:

| Step | Action |
|---|---|
| 1 | Place the cursor at the target position. |
| 2 | Open the function input assistant *(see page 1194)*:<br>● The **Edit → FFB Input Assistant... menu command.**.<br>● The **FFB Input Assistant...menu command**from the shortcut menu (right-click) for the section.<br>● Press the **Ctrl+I**key combination.<br>or<br>● Select the  symbol.<br>**Result:**<br>The function input assistant will be opened.<br>  |

| Step | Action |
|---|---|
| 3 | Click the **Instance** button in the**....column**, to open an FFB selection window *(see page 1184)*.<br>**Result:**<br>The dialog box for selecting function block instances is opened.<br>  |

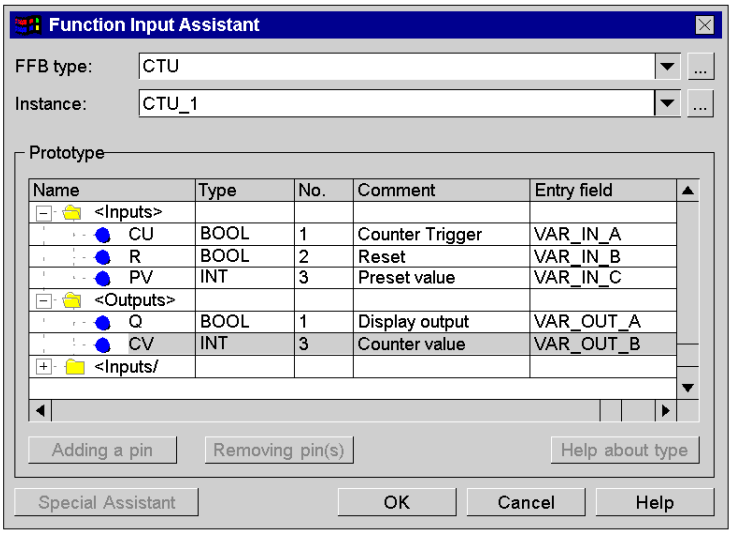
| Step | Action |
|---|---|
| 4 | Select the desired block instance from the **Name** column and confirm the selection with **OK**.<br>**Result:**<br>● The block instance is added to the function input assistant dialog box.<br>● You can see all available parameters *(see Unity Pro, Program Languages and Structure, Reference Manual )* of the function block, see also Information about programming *(see Unity Pro, Program Languages and Structure, Reference Manual )*. |



| Step | Action |
|---|---|
| 5 | Double-click the **Entry field** cell of the first parameter and enter the name of the variable/address to be used.<br>You have the following options to enter the variable/address:<br>● You can now enter the variable name and confirm using **Enter**.<br>● You can select the variable/address from the list of recently used variables/addresses using the ▼ symbol.<br>or<br>● Using the button **...** to open a variable selection dialog box *(see page 1189)*.<br>Assign all parameters of the function block a variable/address in this way.<br>Example:<br>  |
| 6 | Confirm the entries using the **OK** button.<br>**Result:** The function block call is inserted into the ST section and a syntax and semantics check *(see page 1077)* is performed. If the variables are not declared yet a red wavy line under the name is used to identify them.<br>Example:<br>`CTU_1 (CU := VAR_IN_A,`<br>`R := VAR_IN_B,`<br>`PV := VAR_IN_C,`<br>`CV => VAR_OUT_B,`<br>`Q => VAR_OUT_A);` |
| 7 | Declare all variables in the Data Editor *(see page 370)* or directly in the ST section *(see page 1084)*.<br>**Result:** The red wavy line under the variable name disappears. |

## Changing the actual parameter using the function input assistant

Process the following steps to change actual parameters with the function input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor inside a word of the FFB to be changed.<br>Note: No text may be selected and the FFB call to be changed must be syntactically correct. |
| 2 | Open the function input assistant *(see page 1194)*:<br>● The **Edit → FFB Input Assistant... menu command.**.<br>● The **FFB Input Assistant...menu command**from the shortcut menu (right-click) for the section.<br>● Press the **Ctrl+I**key combination.<br>or<br>● Select the  symbol.<br>**Result:**<br>The function input assistant is opened with the current actual parameters.<br>  |
| 3 | Make the desired changes and confirm using **OK**. |



## Inserting FFBs via drag & drop

FFBs can also be inserted into the section via drag & drop using the **Types Library Browser**.

You have the following options to activate the **Types Library Manager**:

- Use the **Tools → Types Library Manager** menu command.
  or
- Press the **Alt+3** key combination.

# Using public variables

## Introduction

In addition to inputs/outputs, some function blocks also provide public variables.

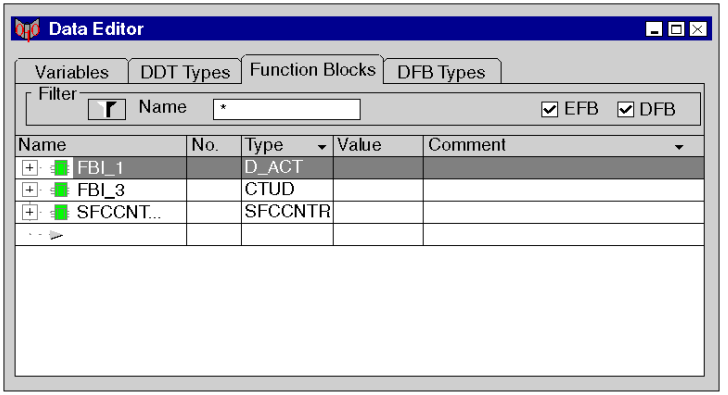
These variables transfer statistical values (values that are not influenced by the process) to the function block. They are used for setting parameters for the function block.

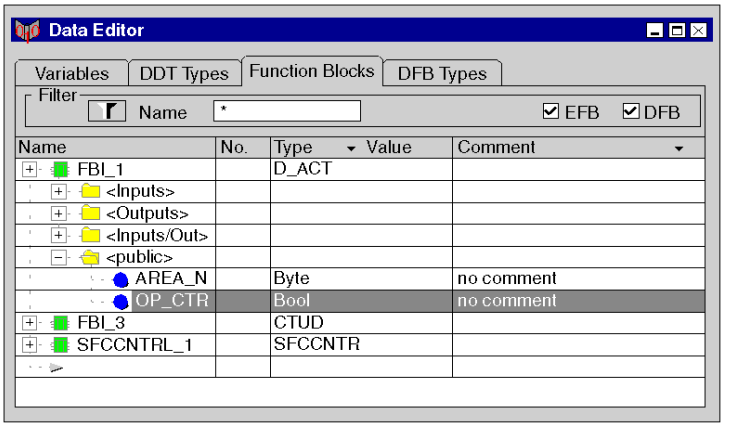
The assignment of values to public variables is made via their initial values or assignments.

Public variables are read via the instance name of the function block and the names of the public variables.

## Assigning values using the initial value

Process the following steps to assign values using the initial value:

| Step | Action |
|---|---|
| 1 | Call the function block for whose public variable you want to assign a value, see also *Calling an FFB, page 1088*. |
| 2 | Open the data editor, see also *Data Editor Access, page 328*. |
| 3 | Select the **Function Block** tab.<br>**Result:**<br>The function block instances are displayed.<br>  |

| Step | Action |
|---|---|
| 4 | Expand the display of the public variables by clicking the respective **+** symbols.<br>**Result:**<br>You can now see all available public variables of the function block.<br>  |
| 5 | Enter the desired value for the public variable in the **Value** box. |

## Assigning values using the assignment operator

Process the following steps to assign values using the assignment operator:

| Step | Action |
|---|---|
| 1 | Call the function block for whose public variable you want to assign a value, see also *Calling an FFB, page 1088*. |
| 2 | Call the public variable, see also *Using formal parameters, page 1085*.<br>For example:<br>`D_ACT1.OP_CTRL` |
| 3 | Assign a value to the public variable.<br>For example:<br>`D_ACT1.OP_CTRL := 1 ;` |

## Reading public variables

See *Using formal parameters, page 1085*

# Calling a statement

## Introduction

You can call a statement (IF, FOR, WHILE, REPEAT, CASE) in the same way that you enter text (syntax, see section Statement *(see Unity Pro, Program Languages and Structure, Reference Manual )* in the Reference manual) or using an input assistant.

## Selecting statements

Selecting statements:

| Statement | Description | Call via menu command | Call via symbol | Call via key combination |
|---|---|---|---|---|
| IF | The IF statement determines that a statement or a group of statements will only be executed if its related Boolean expression has the value 1 (true). If the condition is 0, the statement or the statement group will not be executed. See also in the Reference manual: IF...THEN...END_IF *(see Unity Pro, Program Languages and Structure, Reference Manual )*, ELSE *(see Unity Pro, Program Languages and Structure, Reference Manual )* and ELSIF...THEN *(see Unity Pro, Program Languages and Structure, Reference Manual )* | **Edit → New → IF Statement** |  | **F4** |
| FOR, | The FOR statement repeats a statement sequence until the END_FOR statement. The number of repetitions is determined by the start value, the end value and the control variable. See also in the Reference manual: FOR...TO...BY...DO...END_FOR *(see Unity Pro, Program Languages and Structure, Reference Manual )*, | **Edit → New → FOR Statement** |  | **F5** |

| Statement | Description | Call via menu command | Call via symbol | Call via key combination |
|---|---|---|---|---|
| WHILE | The WHILE statement causes a sequence of statements to be executed repeatedly until its related Boolean expression is 0. If the expression is false right from the start, the group of statements will not be executed at all. See also in the Reference manual: WHILE...DO...END_WHILE *(see Unity Pro, Program Languages and Structure, Reference Manual )*, | **Edit → New → WHILE Statement** |  | **F6** |
| REPEAT | The REPEAT statement causes a sequence of statements to be executed repeatedly (at least once) until its related Boolean condition is 1. See also in the Reference manual: REPEAT...UNTIL...END_REPEAT *(see Unity Pro, Program Languages and Structure, Reference Manual )*, | **Edit → New → REPEAT Statement** |  | **Shift + F6** |
| CASE | The CASE statement consists of an INT data type expression (the "selector") and a list of statement groups. Each group is provided with a label which consists of one or several integer numbers (INT, DINT, UINT, UDINT) or areas of integer values. The first group of statements is executed, whose label contains the calculated value of the selector. Otherwise none of the statements will be executed. See also in the Reference manual: CASE...OF...END_CASE *(see Unity Pro, Program Languages and Structure, Reference Manual )*, | **Edit → New → CASE Statement** |  | **Shift + F4** |

## Calling a IF statement using the input assistant

Process the following steps to call a IF statement with the input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor at the beginning of an empty line. |
| 2 | Select the input assistant for IF statements, see also *Selecting statements, page 1101*.<br>**Result:**<br>An empty IF statement is inserted into the section.<br>`IF THEN`<br>`ELSEIF THEN`<br>`ELSE`<br>`END_IF;` |
| 3 | Complete the IF statement with your data.<br>**Result:** A syntax and semantics check *(see page 1077)* is performed immediately after the statement is entered.<br>**Note:** An input assistant is also provided for entering variables *(see page 1082)* and calling FFBs *(see page 1088)* (functions, function blocks and procedures). For example:<br>`IF A>B THEN`<br>`C:=SIN(A) * COS(B);`<br>`ELSIF A=B THEN`<br>`C:=ADD(A,B);`<br>`ELSE`<br>`B:=MUL(C,A);`<br>`END_IF;` |

## Calling a FOR statement using the input assistant

Process the following steps to call a FOR statement with the input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor at the beginning of an empty line. |
| 2 | Select the input assistant for FOR statements, see also *Selecting statements, page 1101*.<br>**Result:**<br>An empty FOR statement is inserted into the section.<br>`FOR TO BY DO`<br>`END_FOR;` |

| Step | Action |
|---|---|
| 3 | Complete the FOR statement with your data.<br>**Result:** A syntax and semantics check *(see page 1077)* is performed immediately after the statement is entered.<br>**Note:** An input assistant is also provided for entering variables *(see page 1082)* and calling FFBs *(see page 1088)* (functions, function blocks and procedures).<br>For example:<br>`FOR i:= 10 TO 1 BY -1 DO`<br>`C:=SIN(A) * COS(B);`<br>`END_FOR;` |

## Calling a WHILE statement using the input assistant

Process the following steps to call a WHILE statement with the input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor at the beginning of an empty line. |
| 2 | Select the input assistant for WHILE statements, see also *Selecting statements, page 1101*.<br>**Result:**<br>An empty WHILE statement is inserted into the section.<br>`WHILE DO`<br>`END_WHILE;` |
| 3 | Complete the WHILE statement with your data.<br>**Result:** A syntax and semantics check *(see page 1077)* is performed immediately after the statement is entered.<br>**Note:** An input assistant is also provided for entering variables *(see page 1082)* and calling FFBs *(see page 1088)* (functions, function blocks and procedures).<br>For example:<br>`WHILE X <= 100 DO`<br>`  X := X + 4;`<br>`END_WHILE;` |

## Calling a REPEAT statement using the input assistant

Process the following steps to call a REPEAT statement with the input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor at the beginning of an empty line. |
| 2 | Select the input assistant for REPEAT statements, see also *Selecting statements, page 1101*.<br>**Result:**<br>An empty REPEAT statement is inserted into the section.<br>`REPEAT`<br>`UNTIL`<br>`END_REPEAT;` |
| 3 | Complete the REPEAT statement with your data.<br>**Result:** A syntax and semantics check *(see page 1077)* is performed immediately after the statement is entered.<br>**Note:** An input assistant is also provided for entering variables *(see page 1082)* and calling FFBs *(see page 1088)* (functions, function blocks and procedures).<br>For example:<br>`REPEAT`<br>`  x := x + 2;`<br>`UNITL x >= 101`<br>`END_REPEAT;` |

## Calling a CASE statement using the input assistant

Process the following steps to call a CASE statement with the input assistant:

| Step | Action |
|---|---|
| 1 | Place the cursor at the beginning of an empty line. |
| 2 | Select the input assistant for CASE statements, see also *Selecting statements, page 1101*.<br>**Result:**<br>An empty CASE statement is inserted into the section.<br>`CASE  OF`<br>`ELSE`<br>`END_CASE;` |
| 3 | Complete the CASE statement with your data.<br>**Result:** A syntax and semantics check *(see page 1077)* is performed immediately after the statement is entered.<br>**Note:** An input assistant is also provided for entering variables *(see page 1082)* and calling FFBs *(see page 1088)* (functions, function blocks and procedures).<br>For example:<br>`CASE X OF`<br>`1,5:  C:=SIN(A) * COS(B);`<br>`2:  B:=C - A;`<br>`6..10: C:=C * A;`<br>`ELSE`<br>`B:=C * A;`<br>`C:=A / B;`<br>`END_CASE;` |

# Online functions

## Online functions

The description of online functions is found in chapter *Debugging in Textual Languages (Structured Text, Instruction List)*, *page 1431*.

# Reference Data Type in ST

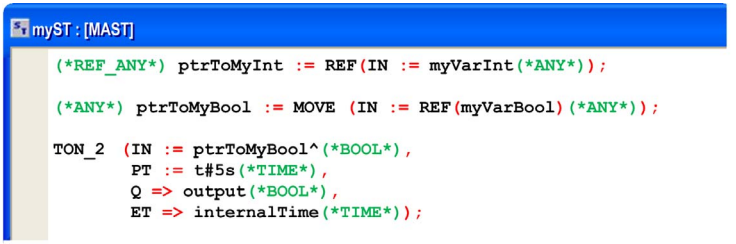
## Introduction

References to variables can be assigned in ST application programs.

A reference can be de-referenced in ST application programs.

For more information, refer to Reference Data Type *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

ST Examples using the Reference Data Type:




The EF, `REF`, takes a variable as an input parameter (type `ANY`) and provides its address as output parameter (type `REF_ANY`).

The EF, `MOVE`, can assign a reference to a variable to another reference. If the input parameter is (`REF(MyVariable)`), `MOVE` has the same effect as the `REF` EF.

The EF, `TON`, takes `ptrToMyBool^` as an input parameter to dereference the reference ptrToMyBool.

# Export/Import IL Sections

## Export/Import

The description for exporting/importing sections is found in the chapter *Import / Export, page 1719*.

# Chapter 27
# LL984 Editor

## Overview

This chapter describes the specifics of the LL984 Editor.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 27.1 | Creating a Program with LL984 Programming Language | 1112 |
| 27.2 | Editing Contacts | 1134 |
| 27.3 | Editing Coils | 1144 |
| 27.4 | Editing Function Blocks | 1152 |
| 27.5 | Editing Links | 1157 |
| 27.6 | Editing Equation Network Blocks | 1163 |
| 27.7 | Assigning Variables to LL984 Objects | 1171 |
| 27.8 | Calling Subroutines | 1175 |
| 27.9 | Entering Comments | 1177 |
| 27.10 | Customize LL984 Editor Colors | 1179 |

# Section 27.1
## Creating a Program with LL984 Programming Language

### Overview

This section describes the basics for creating a program in the LL984 programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Project Settings for LL984 Programs | 1113 |
| Structure of an LL984 Program | 1114 |
| Creating an LL984 Program | 1117 |
| Syntax and Semantics Check During Programming | 1120 |
| Network Display | 1123 |
| Navigating with the Keyboard | 1124 |
| Selecting Objects | 1126 |
| Deleting, Cutting, Copying, Pasting and Moving Objects | 1128 |
| Displaying the Properties | 1131 |
| Inserting and Deleting Rows/Columns | 1132 |

# Project Settings for LL984 Programs

## Introduction

Before creating an LL984 program, you should activate the following project settings:

- **Ladder Logic 984 (LL984)**
  (**Project Settings** → **Program** → **Languages**)
  To be able to create LL984 sections and networks.
- **Allow dynamic arrays (ANY_ARRAY_XXX)**
  (**Project Settings** → **Variables**)
  To be able to use dynamic arrays with the LL984 function blocks.

# Structure of an LL984 Program

## Introduction

The structure of an LL984 program corresponds to a rung for relay switching.

The left power rail is located on the left side of the LL984 Editor. This left power rail corresponds to the phase (L conductor) of a rung.

The right power rail corresponds to the neutral conductor.

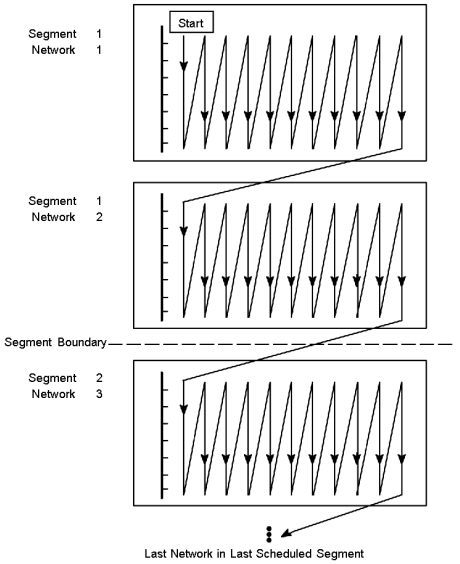
A group of connected objects that are not connected to other objects (except for the power rails), is called a rung.

## Properties of an LL984 program

In LL984 program:

- The logic of LL984 is programmed in networks.
- Several networks are grouped in segments.
- Each network contains a matrix of 7 rows and 11 columns, where you can place contacts, coils, function blocks etc.
- LL984 programs are cell oriented, i.e. only 1 object can be placed in each cell.
  With the exception of vertical connections. They may coexist with other objects (contact, coils, function blocks).
- LL984 segments (and also LL984 networks inside segments) are executed in the sequence in which they are displayed in the **Structural View** of the **Project Browser**. To change this sequence, you can drag-and-drop LL984 segments or LL984 networks inside the **Structural View**.

Objects in a network are executed top to bottom, left to right.




- A syntax and semantics check is performed immediately after the statement is entered. The result of this check is displayed in colored text and objects.
  Please refer to Syntax and Semantics Check During Programming *(see page 1120)*.
- Syntactically or semantically incorrect segments or networks can be saved.

## LL984 Operating Modes

**NOTE:** to detect the first cycle in RUN, the recommended solution is to use **%S21** system bit.

## LL984 Program Objects

The objects of the LL984 programming language help to divide a network into a number of:

- contacts *(see page 1134)*
- coils *(see page 1144)*
- function blocks
- equation network blocks *(see page 1163)*
- jumps and subroutine calls *(see page 1175)*

These objects can be linked with each other through:

- boolean connections and links *(see page 1157)*
- variables

Comments for the logic of the program can be added using text objects *(see page 1177)*.

### Edit and View Functions

- Selecting objects
- Deleting objects
- Cutting, copying and pasting objects
- Moving objects (also between different LL984 sections)
- Replacing objects
- Undo and Redo
- Using Bookmarks *(see page 576)*
- Searching and replacing variables and function blocks *(see page 1235)*

### FFB Input Features

The LL984 Editor provides 2 additional features to add function blocks to a network:

- LL984 Instruction Bar *(see LL984 Editor, Reference Manual, LL984 Specifics)*
- mnemonics *(see LL984 Editor, Reference Manual, LL984 Specifics)*

Using mnemonics *(see page 1156)* you can also add coils and contacts to a network.

### Online Functions

Displaying and modifying variable values using:

- animation tables *(see page 1512)*
- instruction editors *(see page 1522)*

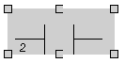
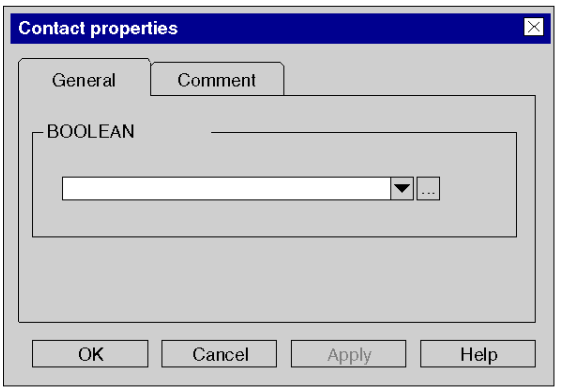
# Creating an LL984 Program

## Introduction

The LL984 Editor window is made up of cells and 1 object can be placed in each one. The cells are separated visually using a grid which can be switched off (**View** → **Grid**).

## Creating an LL984 Program with the Mouse

Process the following steps to create an LL984 program with the mouse:

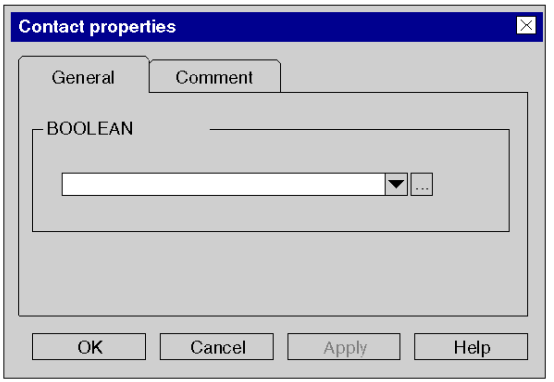
| Step | Action |
|---|---|
| 1 | Create an LL984 network. *(see page 503)* |
| 2 | Select the desired object using:<br>● the menu commands in the **Edit** → **New** → **<Object>** menu<br>or<br>● the symbols for the objects in the toolbar<br>**Result:** The mouse pointer indicates the selected object (placement mode). |
| 3 | Click the target cells in the LL984 section.<br>**Result:** The selected object is inserted and the selection mode is active again.<br>Example:<br>  <br>Return to selection mode by clicking the  symbol or pressing the **ESC** key. |
| 4 | To enter the respective actual parameters (variable/address), double-click the object.<br>**Result:** The properties dialog box for the object is opened.<br>Example:<br>  |

| Step | Action |
|---|---|
| 5 | Enter the desired actual parameter and a comment (if required) and confirm the entries with **OK**.<br>**Result:** The names of the associated variables are displayed above the object.<br>**NOTE:** Entering a binary address opens the **Create variable** dialog. By this you can create a variable associated with the entered address. |
| 6 | Repeat these steps until all objects are entered. |

## Creating an LL984 Program with the Keyboard

Process the following steps to create an LL984 program with the keyboard:

| Step | Action |
|---|---|
| 1 | Create an LL984 network. *(see page 503)* |
| 2 | Using the keyboard, place the field with the gray background on the cell where the object should be inserted. |
| 3 | Select the desired object using the function keys for the objects.<br>**Result:** The cursor symbol indicates the selected object. |
| 4 | Confirm the selection using the **ENTER** key.<br>**Result:** The selected object is inserted into the cell with the gray background and the gray field is automatically shifted to the next cell.<br>Example:<br>  <br><br>To return to selection mode, press the **ESC** key. |

| Step | Action |
|---|---|
| 5 | To enter the respective actual parameters (variable/address), use the **Properties** menu command from the shortcut menu.<br>**Result:** The properties dialog box for the object is opened.<br>Example:<br>  |
| 6 | Enter the desired actual parameter and a comment (if required) and confirm the entries with **OK**.<br>**Result:** The names of the associated variables are displayed above the object.<br>**NOTE:** Entering a binary address opens the **Create variable** dialog. By this you can create a variable associated with the entered address. |
| 7 | Repeat these steps until all objects are entered. |

# Syntax and Semantics Check During Programming

## Introduction

Syntax and semantics check is performed directly when creating the program.

The result of this check is displayed in 3 formats:

- Directly in the program section with colored objects or texts.
- As tooltip if the cursor is placed on an incorrect object.
- In the output window, if **Build** → **Analyze** is selected.

**NOTE:** Warnings are not visualized in the program section.

## Colors of Objects/Texts

| Color | Description |
|---|---|
| black | Syntax and semantics are correct. |
| blue | Possible causes:<br>● Associated variable is not declared.<br>● Data type of the variable does not match the data type of the pin.<br>● Input or output pin not connected. |
| red wavy line below text | invalid text. For example, variables which are not declared, variables with an incorrect data type, etc. |

## Tooltips

If the cursor is placed over an incorrect object, a tooltip is displayed with a brief description of the cause of the error/warning.

## Messages in the Output Window

Error and warning messages are displayed in the output window after **Build** → **Analyze**.

Double-click an error or a warning message in the output window to navigate to the affected object in the LL984 Editor.

Errors messages in contrast to warning messages must be fixed, because they inhibit a successful **Build**.

## Error Messages

The following is a list of example error messages that **Analyze** will display (E).

| Error Message |
|---|
| illegal link |
| variable '%1' must be located |
| only coils allowed in last column |
| only one coil per row |
| FFB type '%1' isn't supported |
| last object in row must be coil |
| maximum column number allowed is '11' |
| maximum row number allowed is '7' |
| parameter '%1' expects constant value |
| illegal constant value '%1': %2 |
| parameter '%1' expects variable |
| unsupported object |
| dimension of '%1' is too small, must be at least '%2' |
| dimension of '%1' must be '%2' |
| variable '%1' must be writable |
| parameter of '%1' must be a multiple of '%2' |
| parameter of '%1' must be a multiple of '%2' |
| usage of LL984 language is disabled |
| usage of SKP instruction is disabled |
| '%1' only allowed in subroutine network |
| '%1' must be in row1/column1 |
| label '%1' isn't unique |
| label '%1' doesn't exist |
| subroutine segment doesn't exist |
| file '%1' is missing |
| IMC template '%1' doesn't exist |
| Non-latched usage of '%1' detected, which is used also as latched |

## Warning Messages

The following is a list of example warning messages that **Analyze** will display (W).

| Warning Message |
|---|
| expression expected (if no actual parameter is assigned to a contact) |
| variable expected (if no actual parameter is assigned to a coil) |

## Tolerated Problems

**Analyze** will tolerate the following problems.

| Problem |
|---|
| missing connections (to power rails or other logic elements) |
| skipping from a subroutine network to a network of another subroutine in the LL984 subroutine segment, if the project setting "SKP enabled" is checked |

# Network Display

## Introduction

The LL984 Editor can display various information concerning variables attached to coils, contacts and function blocks.

This information is displayed above the respective coil, contact or function block node and can be displayed by tooltip, too.

## 4 Different Views

You can define 4 different views (**View 1** to **View 4**) via **Tools** → **Project Settings** → **Program** → **LL984** → **Network Display**.

According to the settings for **View 1** to **View 4** the LL984 Editor will show the following information:

- address (e.g. %M100)
- 984-address (e.g. 400100)
- variable name
- variable comment
- descriptors 1 to 9
- topological address

For detailed information on how to define the different views, please refer to *Project Settings*, section *Program (see LL984 Editor, Reference Manual, LL984 Specifics)*.

## View Switching

You can switch between the different views defined, using **CTRL+W** or using the **Change View Mode** button (  ).

The button's tooltip shows the current view (**View 1, 2, 3 or 4**.

# Navigating with the Keyboard

## Using the Keyboard

The following keys and key combinations are provided for navigation:

| Key combinations | Motion |
|---|---|
| **Left arrow** | Moves the gray field in a cell to the left by one cell and selects the contents of that cell.<br>If the first line is reached, the previous line is automatically selected. |
| **Right arrow** | Moves the gray field in a cell to the right by one cell and selects the contents of that cell.<br>If the last line is reached, the next line is automatically selected. |
| **Up arrow** | Moves the gray field in a cell to the up by one cell and selects the contents of that cell. |
| **Down arrow** | Moves the gray field in a cell to the down by one cell and selects the contents of that cell. |
| **Shift+Left arrow** | Moves the selected object and the gray field to the left by one cell.<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |
| **Shift+Right arrow** | Moves the selected object and the gray field to the right by one cell.<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |
| **Shift+Up arrow** | Moves the selected object and the gray field up by one cell.<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |
| **Shift+Down arrow** | Moves the selected object and the gray field down one cell.<br>This also applies to several selected objects, the gray field must be behind one of the selected objects. |
| **Home** | Positions the gray field in the first column and shows this new location. |
| **End** | Positions the gray field in the last column and shows this new location. |
| **Ctrl+Home** | Positions the gray field in the upper left hand cell in this network and shows this new location. |
| **Ctrl+End** | Positions the gray field in the upper right hand cell in this network and shows this new location. |

| Key combinations | Motion |
|---|---|
| **Page Up** | Scrolls the position of the gray field one page up and shows this new position, as long as the first row is not visible.<br>Displays the previous network/segment, if the first row is already visible. You can also display the previous segment/network via **View → GoTo Previous Section**. |
| **Page Down** | Scrolls the position of the gray field one page down and shows this new position, as long as the last row is not visible.<br>Displays the next network/section, if the last row is already visible. You can also display the next section/network via **View → GoTo Next Section**.<br>Using **Page Down** in the last network of a segment, if the last row is already visible, will display a message box asking you to create a new network.<br>Click **Yes** to open the **New Network** dialog.<br>Click **No** to open the next segment/network. |
| **Space bar** | Selects or deselects the object in the cells highlighted in gray. |
| **Shift** + **Space bar** | The position of the gray field is moved one cell to the right. |
| **Enter** | In Insert mode: Inserts the selected object into the currently selected cells highlighted in gray and moves the position of the gray field one cell to the right. |
| **Alt+Enter** | Opens the properties dialog box for the selected object/pin. |
| **Tab** | Select the next pin in an FFB if the FFB or an FFB pin is selected. |
| **Esc** | Activates the Select mode. |

**NOTE:** You can also display a segment/network by double-clicking the respective segment/network in the **Structural View** of the **Project Browser**.

# Selecting Objects

## Select Mode

Objects are selected in select mode.

Select mode can be activated using:

- the **Edit** → **Select Mode** menu command
- the  symbol,
  or
- the **Esc** key

The  cursor symbol indicates that select mode is active

## Selecting an Object

| Using the Mouse | Using the Keyboard |
|---|---|
| Left-click the object you want to select. | Move the gray field to the cell with the object to be selected, see also Navigating with the Keyboard *(see page 1124)*. |

## Selecting Several Objects

| Using the Mouse | Using the Keyboard |
|---|---|
| **1.** Click the left mouse button and keep it pressed.<br>**2.** Drag the mouse across the objects you want to select.<br>or<br>**1.** Left-click the first object you want to select.<br>**2.** Press the **Ctrl** key and keep it pressed.<br>**3.** Left-click the next object you want to select.<br>**4.** Repeat these steps until all desired objects are selected. | Selecting several objects:<br>**1.** Move the gray field to the cell with the first object to be selected, see also Navigating with the Keyboard *(see page 1124)*.<br>**2.** Press the **Shift+Space bar** key combination.<br>**Result:** The object is added to the current selection and the gray field is shifted one cell to the right.<br>**3.** Repeat these steps until all desired objects are selected. |

## Selecting by Rows

| Using the Mouse | Using the Keyboard |
|---|---|
| Selecting the contents of a row:<br>**1.** Left-click the number of the row you want to select in the vertical ruler.<br><br>Selecting the contents of several lines:<br>**1.** Left-click the number of the first line you want to select in the vertical ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last line you want to select in the vertical ruler. | - |

## Selecting by Columns

| Using the Mouse | Using the Keyboard |
|---|---|
| Selecting the contents of a column:<br>**1.** Left-click the number of the column you want to select in the horizontal ruler.<br><br>Selecting the contents of several columns:<br>**1.** Left-click the number of the first column you want to select in the horizontal ruler.<br>**2.** Press the **Shift** key and keep it pressed.<br>**3.** Left-click the number of the last column you want to select in the horizontal ruler. | - |

## Selecting All

| Using the Mouse | Using the Keyboard |
|---|---|
| Use the **Edit** → **Select All** menu command. | Press the **Ctrl+A** key combination. |

## Deselecting Objects

| Using the Mouse | Using the Keyboard |
|---|---|
| Left-click an empty space in the section. | Move the gray field. |

# Deleting, Cutting, Copying, Pasting and Moving Objects

## Deleting Objects

| Using the Mouse | Using the Keyboard |
|---|---|
| 1. Select *(see page 1126)* the object to be deleted.<br>2. Use the **Edit** → **Delete** menu command. | Deleting the selected object:<br>1. Select *(see page 1126)* the object to be deleted.<br>2. Press the **Del** key.<br>Deleting the object left of the gray field:<br>1. Press the **Backspace** key. |
| **Result:** The selected object is deleted. The actual parameters are deleted together with the object, even if they were not selected explicitly. Boolean links are only deleted if they are selected explicitly. | |

## Cutting Objects

| Using the Mouse | Using the Keyboard |
|---|---|
| 1. Select the object to be cut.<br>2. Use the **Edit** → **Cut** menu command.<br>or<br>Use **Cut** from the context menu (right-click). | 1. Select the object to be cut.<br>2. Press the **Ctrl+X** key combination. |
| **Result:** The selected object is cut from the section and copied to the clipboard. This is also the case for the actual parameters. | |

The cut object can be inserted in any other position (also in another LL984 network).

## Copying Objects to the Clipboard

| Using the Mouse | Using the Keyboard |
|---|---|
| 1. Select the object to be copied.<br>2. Use the **Edit** → **Copy** menu command.<br>or<br>Use **Copy** from the context menu (right-click). | 1. Select the object to be copied.<br>2. Press the **Ctrl+C** key combination. |
| **Result:** The selected object is copied to the clipboard. This is also the case for the actual parameters. | |

The cut object can be inserted in any other position (also in another LL984 network).

## Pasting Objects from the Clipboard

| Using the Mouse | Using the Keyboard |
|---|---|
| 1. Use the menu command **Edit** → **Insert**.<br>or<br>Use **Paste** from the context menu (right-click).<br>2. Left-click the target position. | 1. Move the cursor to the target position pressing **Ctrl** and using the arrow keys.<br>2. Press the **Ctrl+V** key combination.<br>3. Press **Enter**. |
| **NOTE: (for pasting contacts and coils)** If the target position already contains a contact or coil, the existing object is overwritten by the new object.<br>**NOTE:** The following is the behavior of the **Copy**, **Cut** and **Paste** on a function block instance. It only applies to graphical languages as **FBD**, **LD** and **LL984**.<br>• Using the **Paste** function after a **Copy** of an object:<br>As a result of a **Copy**, a new function block instance (**FBI name**) is used. The **Paste** function creates a new FBI whenever it is repeated. Accordingly, the FBI name is incremented.<br>• Using the **Paste** function after a **Cut** of an object:<br>As a result of a **Cut**, the same instance of the function is used. The **Paste** function uses the same FBI whenever it is repeated. Accordingly, the FBI is identical. | |

## Moving Objects

| Using the Mouse | Using the Keyboard |
|---|---|
| 1. Select the object to be moved.<br>2. Position the mouse pointer on the selected object. (On one of the selected objects if several are selected).<br>**Result:** The mouse pointer changes its symbol to  .<br>3. Click the left mouse button and keep it pressed.<br>**Result:** The mouse pointer changes its symbol to  .<br>4. Drag the object to the new position and release the mouse button. | 1. Position the gray field on the object to be moved.<br>2. Press the **Space bar**.<br>3. Press the **Shift** key and keep it pressed.<br>4. Move the object to the target position using the arrow keys. |

| Using the Mouse | Using the Keyboard |
|---|---|
| **Result:** The selected object is moved from its original position to the target position. The object's actual parameters are moved together with the object. Boolean links are only kept for horizontal movement operations.<br>**NOTE:** A moving of FFBs to cells that are already occupied (e.g. by contact, coil, other FFB, boolean link), is not possible.<br>**NOTE:** The move operation is also possible across sections between different opened LL984 networks.<br>**NOTE:** When objects are moved beyond the visible area of the editor window, automatic scrolling of the window will not begin until you reach the edge of the window with the mouse pointer, not as soon as the objects are moved beyond the edge. | |

## Copying Objects via Drag-and-Drop

| Using the Mouse | Using the Keyboard |
|---|---|
| **1.** Select the object to be copied.<br>**2.** Position the mouse pointer on the selected object.<br>**Result:** The mouse pointer changes its symbol to  .<br>**3.** Click the left mouse button and keep it pressed.<br>**4.** Press the **Ctrl** key and keep it pressed.<br>**Result:** The mouse pointer changes its symbol to  .<br>**5.** Drag the object to the new position.<br>**NOTE:** This is also possible across sections between different opened LL984 networks.<br>**6.** Release the mouse button.<br>**Result:** A copy of the selected object is pasted at the target position. The object's actual parameters (variable/address) are copied together with the object. A new instance is automatically created for FFBs. | - |

# Displaying the Properties

## Displaying the Object Properties

You have the following options to display the object properties dialog:

- Double-click the object.
- Select *(see page 1126)* the object and use the **Edit → Properties...** dialog.
- Select the object and use **Properties** from the context menu (right-click).
- Select the object and then press the **Alt+Enter** key combination.

When the properties dialog is called without an object being selected, the Network Properties *(see page 510)* dialog is opened.

If several objects are selected, the commands do not work.

## Displaying the Data Properties

You have the following options to display the **Data Properties** dialog *(see page 1197)*:

- Via the LL984 network:
  Select an object and use **Data Properties** from the context menu or press the **Ctrl+Enter** key combination.
- Via the Data Editor
  Select the line of the respective object in the Data Editor and use **Data Properties** from the context menu or press the **Ctrl+Enter** key combination or use the **Edit → Data Properties** dialog.

# Inserting and Deleting Rows/Columns

## Introduction

Each network contains a matrix of 7 rows and 11 columns.

But via **Insert/Delete row** and **Insert/Delete column** you can shift rows downwards/upwards and shift columns to the right/left.

## Rules for Inserting Rows/Columns

The following rules apply when inserting rows/columns.

- The new row is inserted above the first selected row.
  The content of the following rows will be shifted to the bottom.
- The new column is inserted left of the first selected column.
  The content of the following columns will be shifted to the right.
- If there are no free rows/columns at the bottom/right of the network, the menu items will be disabled.
- If the new row would intersect a middle or a bottom node of an FFB, the menu items will be disabled.

## Rules for Deleting Rows/Columns

The following rules apply when deleting rows/columns.

- You can not delete a row/column containing any object.
- If there are no free rows/columns at the current position, the menu items will be disabled.

## Inserting Rows

| Step | Action |
|---|---|
| 1 | Select the desired row in the vertical ruler (click the row number). |
| 2 | Use the **Insert row** menu command from the context menu of the vertical ruler or use the **Edit** → **Insert row** menu command.<br>**Result:** A row is inserted above the selected row. |

## Inserting Columns

| Step | Action |
|---|---|
| 1 | Select the desired column in the horizontal ruler (click the column number). |
| 2 | Use the **Insert column** menu command from the context menu of the horizontal ruler or use the **Edit** → **Insert column** menu command.<br>**Result:** A column is inserted left of the selected column. |

## Deleting Rows

| Step | Action |
|---|---|
| 1 | Select the desired row in the vertical ruler (click the row number). |
| 2 | Use the **Delete row** menu command from the context menu of the vertical ruler or use the **Edit** → **Delete row** menu command.<br>**Result:** The row is deleted. The rows below are shifted on row up. At the bottom an empty row is inserted. |

## Deleting Columns

| Step | Action |
|---|---|
| 1 | Select the desired column in the horizontal ruler (click the column number). |
| 2 | Use the **Delete column** menu command from the context menu of the horizontal ruler or use the **Edit** → **Delete column** menu command.<br>**Result:** The column is deleted. The columns from the right are shifted one column to the left. At the right side of the grid an empty column is inserted. |

# Section 27.2
# Editing Contacts

## Overview

This section describes how to edit contacts in the LL984 programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selecting Contacts | 1135 |
| Placing Contacts | 1137 |
| Properties Dialog for Contacts | 1141 |

# Selecting Contacts

## Introduction

A contact is an LL984 element that transfers a status on the horizontal link to its right side. This status comes from the Boolean AND link of the status of the horizontal link on the left side with the status of the relevant Boolean current parameter.

## Information Concerning Variables

The LL984 Editor can display various information concerning variables attached to coils, contacts and function block nodes.

This information can be displayed by tooltip, too.

For further information on how the variables attached to contacts are displayed, please refer to *Network Display (see page 1123)*.

## Contact Types

| Contact Type | Description | Call via Menu Command | Call via Icon | Call via Key |
|---|---|---|---|---|
| Normally open contact | For normally open contacts, the status of the left link is copied to the right link, if the status of the relevant Boolean actual parameter is 1. Otherwise, the status of the right link is 0. | **Edit → New → Normally open contact** or **Normally open contact** from the context menu |  | **F3** |
| Normally closed contact | For normally closed contacts, the status of the left link is copied to the right link, if the status of the relevant Boolean actual parameter is 0. Otherwise, the status of the right link is 0. | **Edit → New → Normally closed contact** or **Normally closed contact** from the context menu |  | **Shift+F3** |
| Positive transition-sensing contact | With contacts for detection of positive transitions, the right link for a program cycle is 1 if a transfer of the relevant actual parameter goes from 0 to 1 and the status of the left link is 1 at the same time. Otherwise, the status of the right link is 0.<br>Also see Edge Recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | **Edit → New → Positive transition-sensing contact** or **Positive transition-sensing contact** from the context menu |  | **Ctrl+F3** |

| Contact Type | Description | Call via Menu Command | Call via Icon | Call via Key |
|---|---|---|---|---|
| Negative transition-sensing contact | With contacts for detection of negative transitions, the right link for a program cycle is 1 if a transfer of the relevant actual parameter goes from 1 to 0 and the status of the left link is 1 at the same time. Otherwise, the status of the right link is 0.<br>Also see Edge Recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )*. | **Edit → New → Negative transition-sensing contact** or **Negative transition-sensing contact** from the context menu |  | **Ctrl** + **Shift** + **F3** |

# Placing Contacts

## Introduction

Contacts can be placed in any free cell except cells directly on the right power rail.

If a contact is placed in a cell that was previously occupied by a contact, coil, a Boolean link or a label, the cell contents are replaced by the new contact.

If a contact is placed in a cell that is already occupied by another object (e.g. FFBs), an error message is returned.

The following are permitted as actual parameters for contacts:

- Boolean variables
- Boolean literals (`0`, `1`, `FALSE`, `TRUE`)
- Boolean addresses (topological addresses or symbolic addresses)
- ST expressions that return a Boolean result (e.g. `VarA > VarB`)
  ST expressions as formal parameters on contacts are an extension of IEC 61131-3 and must be explicitly enabled via the **Tools → Project Settings → Program → Languages → Common** by activating the **Usage of ST expressions** check box.

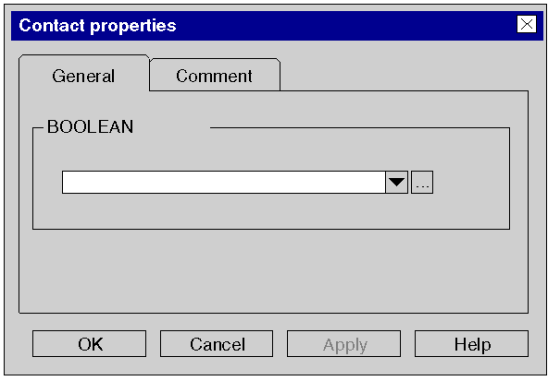
## Automatically Assign a Variable

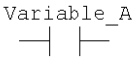
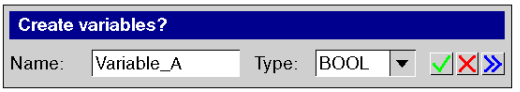
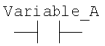
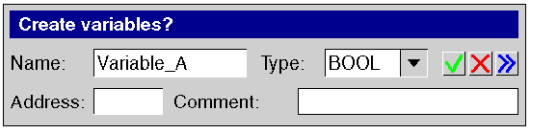
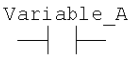
If via **Tools → Options → Data and Languages → Languages** the **Automatically assign a variable to a new graphical object** check box is activated, then the corresponding properties dialog is automatically opened when an object is placed. The procedures explained here are used when the check box is **deactivated**.

## Placing Contacts

| Step | Action |
|---|---|
| 1 | Select the desired contact, see also *Selecting Contacts* *(see page 1135)*. |
| 2 | Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position, and press **Enter**.<br>**Result:** The selected contact is inserted. |
| 3 | To place further contacts of the same type:<br>● Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Assigning Actual Parameters

| Step | Action |
|---|---|
| 1 | Open the properties dialog *(see page 1131)* for the contact.<br>**Result**: The contacts properties dialog is opened.<br>Example:<br>  |
| 2 | You have the following options to enter the actual parameters:<br>● You can enter %Mx (%Ix) and confirm using **Enter**.<br>● You can enter the actual parameter and confirm using **Enter**.<br>● You can select the actual parameter from the list of recently used names using the ▾ symbol and confirm using **Enter**.<br>or<br>● Using the button **...** you can open a **Data Selection** dialog and confirm the selected variables there with **OK**. |

| Step | Action |
|---|---|
| 3 | Confirm the selected variable with **OK**.<br>**Result:**<br>● If the selected variable is declared, it is enabled in the section. A syntax and semantics check *(see page 1120)* is performed and the procedure ends here.<br>  <br>● If the variable is not declared, a **Create variable?** dialog opens. In this case, continue with the following steps of this procedure.<br>  <br>**NOTE:** If you wish to declare the variable later, close the dialog using the symbol. In this case, the variable name is enabled in the section, but not declared.<br>**NOTE:** Entering a binary address (%M1 / %I1) opens the **Create variable** dialog proposing a variable name (_000001 / _100001) that can be modified. If there is already a variable associated to %M1 / %I1 it will be reused as proposal. |
| 4 | If you do not want to assign the variable an address or comment, confirm using **Enter** or the symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check is performed and the procedure ends here.<br>Example:<br>  |
| 5 | If you do want to assign the variable an address and/or comment, use the symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 6 | Confirm with **Enter** or the symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check is performed immediately after the statement is entered.<br>Example:<br>  |

### Usage of ST Expressions

To enter ST expressions for contacts the same rules apply as for entering ST expressions for FFB pins *(see page 798)*.

# Properties Dialog for Contacts

## Calling the Properties Dialog

See Displaying the Properties *(see page 1131)*.

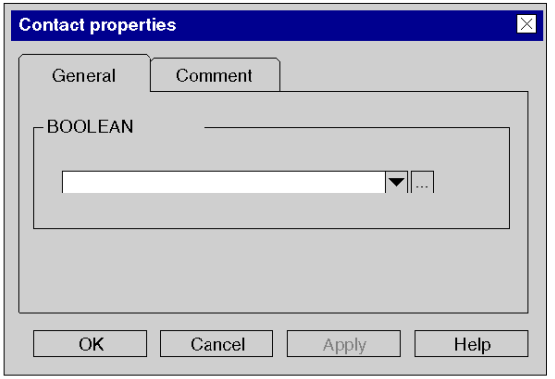
## Structure of the Properties Dialog

The contact properties dialog consists of 2 tabs:

- **General**
  In this tab, you can enter the actual parameter *(see page 1138)* of the contact.
- **Comment**
  In this tab, you can enter a comment about the contact.

## General Tab

Representation of the **General** tab



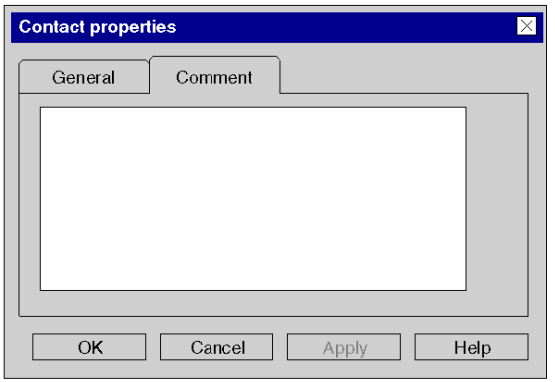
Elements of the **General** tab

| Element | Description |
|---|---|
| **BOOLEAN Expression** | Enter the name of the actual parameter in this text box.<br>You have the following options:<br>● You can directly enter the name of the variable/address or paste it from the clipboard.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Use the command button **...**to open a **Data Selection** dialog *(see page 1182)*.<br><br>The actual parameters which are permitted are:<br>● Boolean variables<br>● Boolean constants<br>● Boolean addresses (topological addresses or symbolic addresses)<br>● ST expression delivering a Boolean result (e.g. `BoolVar1 OR BoolVar1`)<br>● Literal (`1` or `0` or. `TRUE` or `FALSE`) |
| **OK** | Use this command button to accept the entries and close the dialog. |
| **Apply** | Use this command button to accept the entries without closing the properties dialog. |

**NOTE:** Entering a binary address (%M1 / %I1) opens the **Create variable?** dialog proposing a variable name (_000001 / _100001) that can be modified. If there is already a variable associated to %M1 / %I1 it will be reused as proposed.

## Comment Tab

Representation of the **Comment** tab



Elements of the **Comment** tab

| Element | Description |
|---|---|
| Text box | Enter a comment about the contact.<br>This comment is displayed as a tooltip when the cursor is placed over the contact. It is independent from the variable comment that can be assigned to the variable in the Data Editor. |
| **OK** | Use this command button to accept the entries and close the dialog. |
| **Apply** | Use this command button to accept the entries without closing the properties dialog. |

# Section 27.3
# Editing Coils

## Overview

This section describes how to edit coils in the LL984 programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selecting Coils | 1145 |
| Placing Coils | 1147 |
| Property Dialog for Coils | 1150 |

# Selecting Coils

## Introduction

A coil is an LL984 element which transfers the status of the horizontal link on the left side, unchanged, to the horizontal link on the right side. The status is saved in the respective Boolean actual parameter. Coils normally follow contacts or FFBs, (functions and function blocks), but they can also be followed by contacts.

## Information Concerning Variables

The LL984 Editor can display various information concerning variables attached to coils, contacts and function block nodes.

This information can be displayed by tooltip, too.

For further information on how the variables attached to coils are displayed, please refer to *Network Display* *(see page 1123)*.

## Coil Types

**⚠ CAUTION**

**UNINTENDED EQUIPMENT OPERATION**

Be sure of the output states after a PLC start when using latched and non-latched coils.

Before using non-latched or latched coils, make sure that you completely understand the different behavior of non-latched/latched coils.

- **Non-latched Coils**
  A reset of the coil will be performed during warm-start and after STOP/RUN of the PLC. Reset during warm-start works for %M addresses and %M-located-variables attached only.
- **Latched Coils**
  In the first scan latched coils have the status of last scan before warm-start.

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

| Coil Type | Description | Call via Menu Command | Call via Icon | Call via Keyboard |
|---|---|---|---|---|
| Coil (non-latched) | With coils, the status of the left link is copied to the relevant Boolean actual parameter and the right link.<br>For further information see below. | **Edit** → **New** → **Coil**<br>or<br>**Coil** from the context menu |  | **F5** |
| Latched coil | With latched coils, the status of the left link is copied to the relevant Boolean actual parameter and the right link.<br>For further information see below. | **Edit** → **New** → **Latched coil**<br>or<br>**Latched coil** from the context menu |  | **Shift+F5** |

## PLC Start Behavior of Coils

- **Non-latched Coils**
  A reset of the coil will be performed during warm-start and after STOP/RUN of the PLC. Reset during warm-start works for %M addresses and %M-located-variables attached only.
- **Latched Coils**
  Latched coils will maintain their state after a Stop/Start of the PLC and on warm-start. In the first scan latched coils have the status of last scan before warm-start.
  This is the same behavior as normal coils show in IEC Ladder Diagram language.

Detailed information:

- For details when a warm-start is performed please see *Processing of Power Outage and Restoral for Premium/Quantum PLCs (see Unity Pro, Program Languages and Structure, Reference Manual )*.
  Especially for the case of a Power Cycle this depends on the position of the PLC's key switch (**MemPrt**/**Start**/**Stop**).
  - For firmware version 2.8:
    Latched coils are **not** maintained if the key switch is in the Start position.
  - For firmware version 2.6:
    Latched coils **are** maintained regardless of the key switch position.
- For details regarding the PLC's key switch refer to *Key Switches (see Quantum with Unity Pro, Hardware, Reference Manual)*.
  For details regarding Cold Start and warm-start processing refer to *Processing on Cold Start for Premium/Quantum PLCs (see Unity Pro, Program Languages and Structure, Reference Manual )* and *Processing on Warm Restart for Premium/Quantum PLCs (see Unity Pro, Program Languages and Structure, Reference Manual )*.

## Non-latched Coil Restriction

The following restriction applies for non-latched coils:

- If an address (or a variable located on this address) is attached to a normal LL984 coil (non-latched) and this address has other (latched) write usages somewhere else in the program (for example an assignment in ST/FBD/LD or a latched coil in an LL984 network), this will lead to the following analyze message:
  `Non-latched usage of '%1' detected, which is used also as latched.`
  This message will be reported for the non-latched coil(s) usage in LL984 networks only.
  To help avoid this restriction, start a search for getting the occurrences of this conflicting address.

## Latched Coil Representation

A latched coil is represented by a circle with an L or an M in the middle.

Whether an **L** or an **M** will be displayed inside a latched coil can be define via **Tools** → **Options** → **Data and Languages** → **LL984** → **Show latched coil as**. Please refer to *Data and Languages (see LL984 Editor, Reference Manual, LL984 Specifics)*.

# Placing Coils

## Introduction

For placing coils the following restrictions apply:

- Only 1 coil can be placed per row.
- No other object can be placed to the right of a coil. The coil is always the last object in a row.
- If a coil is placed in a cell that is already occupied by another object (e.g. FFBs), an error message is returned.

A placed coil automatically creates a connection with its neighboring objects on the left if they are of the `BOOL` data type, even if free cells are between them.

The following are permitted as actual parameters for coils:

- Boolean variables
- Boolean addresses (topological addresses or symbolic addresses)

## Show Coils in Last Column

Using the main menu **View** you can activate the **Show coil in last column** setting.

If this setting is activated, each coil will be displayed in the last column with a dotted line between its "real" location cell and the last column.

Otherwise each coil will be displayed at its "real" location cell.

**NOTE:** Activating this setting will switch the LL984 Editor to **Read only** mode. No modifications are allowed inside the LL984 Editor. To leave the **Read only** mode, just deactivate this setting.

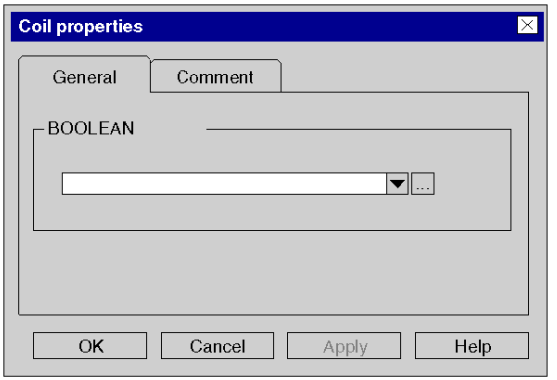
## Automatically Assign a Variable

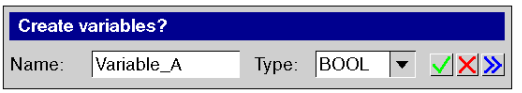
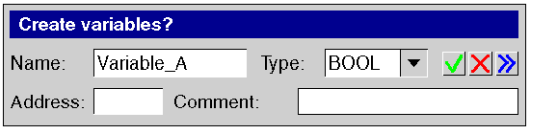
If via **Tools** → **Options** → **Data and Languages** → **Languages** the **Automatically assign a variable to a new graphical object** check box is activated, then the corresponding properties dialog is automatically opened when an object is placed. The procedures explained here are used when the check box is **deactivated**.

## Placing Coils

| Step | Action |
|---|---|
| 1 | Select the desired coil. See also Selecting Coils *(see page 1145)*. |
| 2 | Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position, and press **Enter**.<br>**Result:** The selected coil is inserted. |
| 3 | To place further coils of the same type:<br>● Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Assigning Actual Parameters

| Step | Action |
|---|---|
| 1 | Open the properties dialog *(see page 1131)* for the coil.<br>**Result**: The coils properties dialog is opened.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | You have the following options to enter the actual parameters:<br>● You can enter %Mx (%Ix) and confirm using **Enter**.<br>● You can enter the variable/address name and confirm using **Enter**.<br>● You can select the variable name from the list of recently used names using the ▾ symbol and confirm using **Enter**.<br>or<br>● Using the button **...** you can open a **Data Selection** dialog and confirm the selected variables there with **OK**. |
| 3 | Confirm the selected variable with **OK**.<br>**Result:**<br>● If the selected variable is declared, it is enabled in the section. A syntax and semantics check *(see page 1120)* is performed and the procedure ends here.<br>● If the variable is not declared, a **Create variable?** dialog opens. In this case, continue with the following steps of this procedure.<br>  <br>**NOTE:** If you wish to declare the variable later, close the dialog using the ✕ symbol. In this case, the variable name is enabled in the section, but not declared.<br>**NOTE:** Entering a binary address (%M1 / %I1) opens the **Create variable?** dialog proposing a variable name (_000001 / _100001) that can be modified. If there is already a variable associated to %M1 / %I1 it will be reused as proposal. |
| 4 | If you do not want to assign the variable an address or comment, confirm using **Enter** or the ✓ symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check is performed and the procedure ends here. |
| 5 | If you do want to assign the variable an address and/or comment, use the » symbol for obtaining an advanced dialog and enter the address and/or comment.<br>  |
| 6 | Confirm with **Enter** or the ✓ symbol.<br>**Result:** The variable is declared and enabled in the section. A syntax and semantics check is performed. |

# Property Dialog for Coils

## Calling the Properties Dialog

See Displaying the Properties *(see page 1131)*.

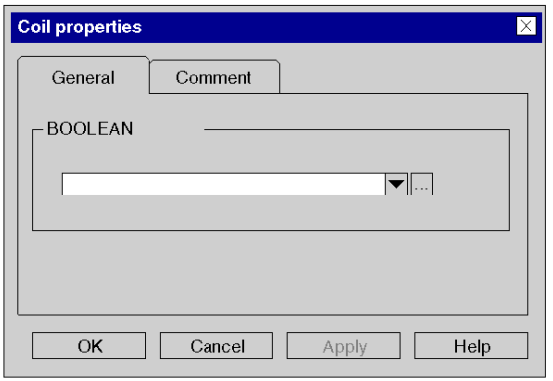
## Structure of the Properties Dialog

The coil properties dialog consists of 2 tabs:

- **General**
  In this tab, you can enter the actual parameter *(see page 1148)* of the coil.
- **Comment**
  In this tab, you can enter a comment about the coil.

## General Tab

Representation of the **General** tab



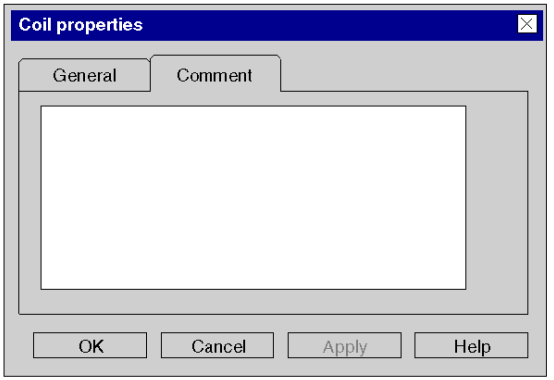

Elements of the **General** tab

| Element | Description |
|---|---|
| **BOOLEAN Expression** | Enter the name of the actual parameter in this text box.<br>You have the following options:<br>● You can directly enter the name of the variable/address or paste it from the clipboard.<br>● You can select the name of the variable from the list of recently used names using the ▼ symbol.<br>or<br>● Use the command button **...** to open a **Data Selection** dialog *(see page 1182)*.<br><br>The actual parameters which are permitted are:<br>● Boolean variables<br>● Boolean addresses (topological addresses or symbolic addresses) |

| Element | Description |
|---|---|
| **OK** | Use this command button to accept the entries and close the dialog box. |
| **Apply** | Use this command button to accept the entries without closing the properties dialog box. |

## Comment Tab

Representation of the **Comment** tab




Elements of the **Comment** tab

| Element | Description |
|---|---|
| Text box | Enter a comment about the coil.<br>This comment is displayed as a tooltip when the cursor is placed over the coil. It is independent from the variable comment that can be assigned to the variable in the Data Editor. |
| **OK** | Use this command button to accept the entries and close the dialog box. |
| **Apply** | Use this command button to accept the entries without closing the properties dialog box. |

# Section 27.4
## Editing Function Blocks

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Editing Function Blocks | 1153 |
| Using Mnemonics | 1156 |

# Editing Function Blocks

## Introduction

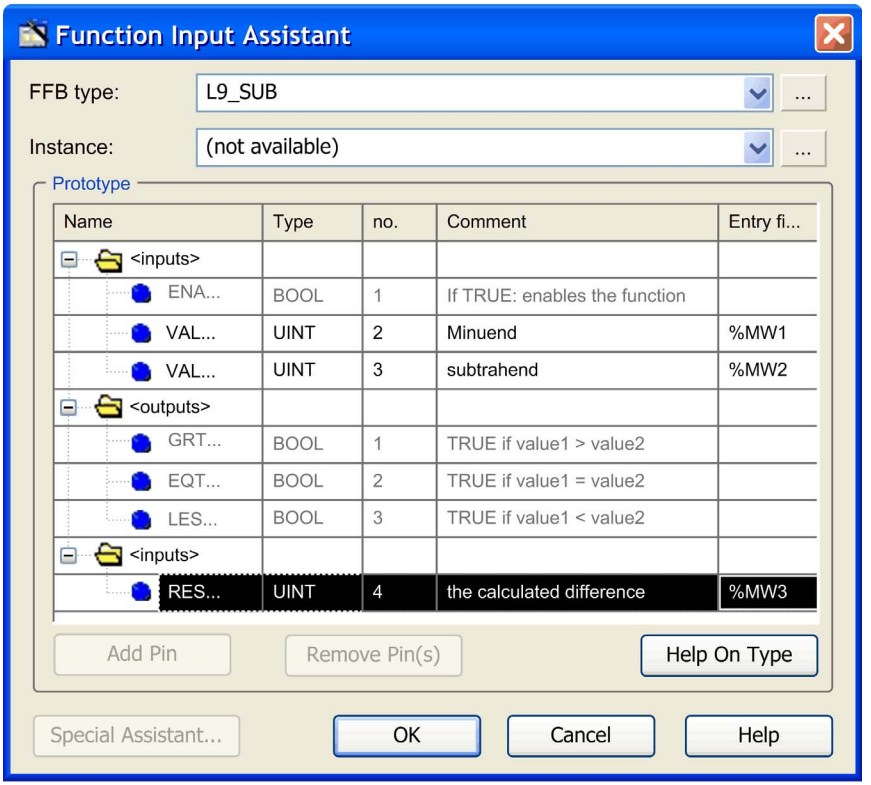
You can insert function blocks using one of the following features:

- **LL984 Instruction Bar**
- **Mnemonics**
- **Data Selection**
- **FFB Input Assistant**.
- **Types Library Browser**,
- **Copy/Paste**

## Inserting a Function Block

| Step | Action |
|---|---|
| 1 | In the LL984 Editor right-click and select from the context menu either<br>● **Data Selection** (**Ctrl+D**) or<br>● **FFB Input Assistant** (**Ctrl+I**).<br>You can also insert a function block using:<br>● drag-and-drop from the **Types Library Browser** (**Alt+3**), .<br>● copy/paste<br>● LL984 Instruction Bar<br>● Mnemonics |
| 2 | Enter the complete name of the function block (e.g. L9_SUB). |
| 3 | Confirm with **Enter**. |
| 4 | Click the cell where you want to insert the function block.<br>**Result**:<br>The function block is inserted (if allowed at this position). |

## Representation of the FFB Input Assistant




You can also see the **Comments** displayed in the dialog above as a tooltip when the cursor is placed over the respective pin of the function block.

## Editing Values

To open the **FFB Input Assistant** for an already inserted function block, right-click the function block and select **FFB Input Assistant** from the context menu.

Click the **Entry Field** and enter a value.

You are only allowed to edit the **Entry fields** displayed in black. Only the nodes (top, middle, bottom) of a function block can be edited.

You can not edit the **Entry fields** displayed in grey (disabled).

## Restrictions

For inserting function blocks the following restrictions apply:

- Only LL984 library function blocks can be inserted in LL984 networks. If you try to insert a function block of another library a message will inform you, that this is not possible.
- You can not insert self-programmed DFBs in LL984 networks.
- LL984 library function blocks can not be inserted in IEC languages sections (FBD, LD, ST, IL). SFC does not use function blocks.
  - FBD and LD editors will refuse insertion and will display a message that insertion is not possible.
  - For textual languages (ST, IL) the call of an LL984 library function block will be detected during **Analyze**.

# Using Mnemonics

## Introduction

Instead of inserting LL984 objects via toolbar, **FFB Input Assistant**, **Data Selection** or drag-and-drop, you can use so called **Mnemonics**.

## Mnemonics Option

Open the **Tools** → **Options** dialog.

Under **Data and Languages** → **Languages** → **LL984** → **Mnemonics** the **Property label** column lists the contacts and coils and an assortment of frequently used functions blocks available for LL984 networks.

In the **Property value** column you can enter a mnemonic for each listed element.

A mnemonic can consist of up to 4 alphanumeric signs.

## Using Mnemonics

As you start typing text, while the current cell marker is on an empty cell in the LL9894 Editor, a small edit box opens in the editor.

Entering a mnemonic (first key typed will be taken over into the edit box) and pressing the **Enter** key will insert the corresponding LL984 element at the current cell.

If the cells needed to insert the element are already occupied, a message box will inform you and the element will not be inserted.

# Section 27.5
# Editing Links

## Overview

This section describes how to edit links in the LL984 programming language.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Selecting Links | 1158 |
| Combining Links | 1159 |
| Placing Links | 1160 |
| Editing Links | 1162 |

# Selecting Links

## Introduction

Links are connections between LL984 objects (contacts, coils, FFBs etc.).

Boolean connections comprise one or more cell-segments that connect Boolean objects (contacts, coils) with one another.

- **Boolean Connection** (horizontal connection)
  This horizontal connection enables series switching of contacts and coils.
  The cell-segments of this connection can be created individually or as a complex connection comprising several cell-segments.
- **Boolean Link** (horizontal link)
  This horizontal link automatically creates a connection between neighboring Boolean objects to the left and right or between an object and the power rail.
- **Vertical Connection**
  This vertical connection enables parallel switching of contacts and coils.

## Link Types

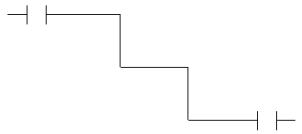
| Link Type | Description | Call via Menu Command | Call via Icon | Call via Key |
|---|---|---|---|---|
| Horizontal connection | horizontal connection, 1 cell wide | **Edit → New → Boolean Connection**<br>or<br>**Boolean Connection** from the context menu |  | **F7** |
| Horizontal link | A horizontal link creates a connection between neighboring Boolean objects to the left and right.<br>If there are no neighboring Boolean objects available, a connection to the power rail is made. | **Edit → New → Boolean Link**<br>or<br>**Boolean Link** from the context menu |  | **Alt+F6** |
| Vertical connection | vertical connection, 1 cell high | **Edit → New → Vertical Connection**<br>or<br>**Vertical Connection** from the context menu |  | **Shift+F7** |

# Combining Links

## Boolean Links

Horizontal and vertical Boolean links can be combined in any way.

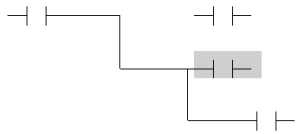
Example:



## Contacts and Coils

Contacts and coils link automatically to existing horizontal or vertical links if no free cells are found between them.

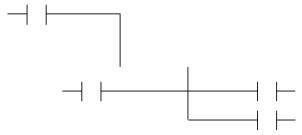
Example:



## Crossing Boolean Links

If 2 Boolean links cross each other they are automatically connected. Since Boolean links may not cross each other there is no special label for them.

Example:



# Placing Links

## Placing Horizontal Connections

Process the following steps to place a horizontal link.

| Step | Action |
|---|---|
| 1 | Select the horizontal connection (**Boolean Connection**). See also Selecting Links *(see page 1158)*. |
| 2 | Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>**Result:** The link is inserted. |
| 3 | To place further links of the same type:<br>● Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Placing Horizontal Links

Process the following steps to place a horizontal link.

| Step | Action |
|---|---|
| 1 | Select the horizontal link (**Boolean Link**). See also Selecting Links *(see page 1158)*. |
| 2 | Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>**Result:** The link is inserted and a connection is created to the neighboring Boolean objects to the left and right.<br>If there are no neighboring Boolean objects available, a connection to the power rail is made. |
| 3 | To place further links of the same type:<br>● Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

## Placing Vertical Connections

On the right side of the cells, there is space to create a vertical connection to the cell below or above. Process the following steps to place a vertical connection.

| Step | Action |
|---|---|
| 1 | Select the vertical connection (**Vertical Connection**). See also Selecting Links *(see page 1158)*. |
| 2 | Click the object which should be connected with the cell below.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>**Result:** The link is inserted. |
| 3 | To place further links of the same type:<br>● Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Editing Links

## Introduction

If you combine several horizontal links this will result in 1 horizontal link.

If you combine several vertical links this will result in 1 vertical link.

If you combine a horizontal and a vertical link this will result in a link made of 2 objects, a horizontal and a vertical link, that can be selected independent from each other.

## Deleting, Cutting, Copying, Pasting and Moving Links

You can delete, cut, copy, paste and move links.

For further information please refer to Deleting, Cutting, Copying, Pasting and Moving Objects *(see page 1128)*.

## Modifying the Size of Links

| Step | Action |
|---|---|
| 1 | Select a link.<br>**Result:** Size handles are displayed at the beginning and at the end of the link. |
| 2 | Click a size handle with the left mouse button and keep it pressed. |
| 3 | Drag the size handle to the desired new end position of the link and release the mouse button. |

# Section 27.6
## Editing Equation Network Blocks

### Overview

This section describes how to edit Equation Network Blocks in the LL984 programming language.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Editing Equation Network Blocks | 1164 |
| Editing the ST Assignment Expression | 1167 |

# Editing Equation Network Blocks

## Introduction

Equation networks in the LL984 Editor are provided as **Equation Network Blocks**.

Equation network blocks offer an opportunity to program complex mathematical functions, with values stored in memory words.

An **Equation Network Block** contains an assignment expression written in ST language.

In general you can use the features provided by ST language as long as it is a valid ST assignment expression.

Please refer to the Structured Text (ST) chapter *(see Unity Pro, Program Languages and Structure, Reference Manual )* and the ST Editor chapter *(see page 1073)*.

## Using the Equation Network Block

You can insert an **Equation Network Block** using the respective toolbar button, via menu (**Edit** → **New** → **Equation Network block**) or via the editors context menu (right mouse click).

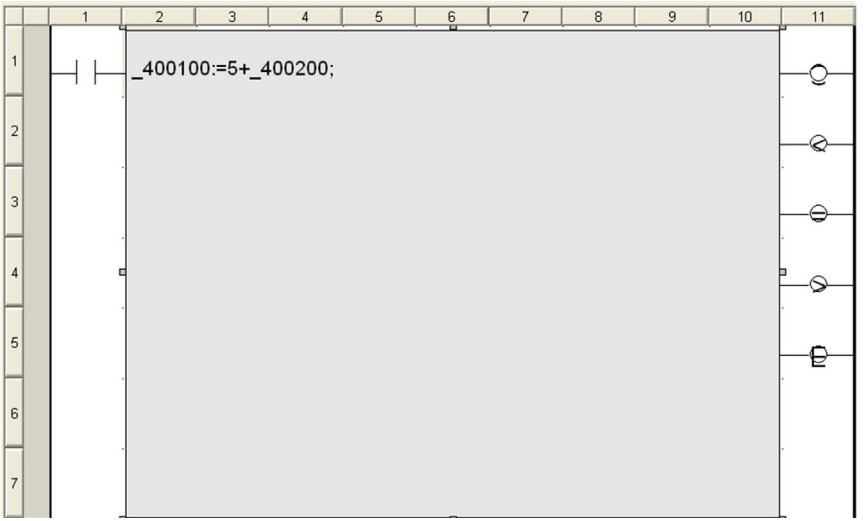
An **Equation Network Block** can only be inserted in an LL984 network at row 1, column 2. The block is 7 rows high and 9 columns wide.

On the left side of an equation network block there is an **Enable** input pin, where you can connect a contact or a boolean connection.

On the right side there are 5 output pins (**OK**, **Lower**, **Equal**, **Greater**,**Error**), where you can connect 5 normally open coils. After inserting these coils are displayed with symbols inside according to the pin names (**O**, **<**, **=**, **>**, **E**).

## Representation

Representation of the **Equation Network Block**




Items of the **Equation Network Block**

| Item | Description |
|---|---|
| input pin **Enable** | you can connect a contact or a boolean connection to enable the block |
| equation network block | expression written in ST language |
| output pins | **OK (O)**: set when the equation is solved without problems |
| | **Lower (<)**: set when the equation result is less than zero |
| | **Equal (=)**set when the equation result is equal to zero |
| | **Greater (>)**set when the equation result is greater than zero |
| | **Error (E)**: set when a problem occurred during solving the equation (see table below) |

## Problems During Solving

| Problem | Description |
|---|---|
| invalid operation | An internal message generated by the math coprocessor. |
| overflow | A value is too large to be represented in its specified data type. |
| underflow | A number is too small to be represented in FP format (for floating point data only). |
| divided by 0 | The variable, constant, or result of a function directly to the right of a / operator has a value of zero. |

# Editing the ST Assignment Expression

## Introduction

After inserting an **Equation Network Block** double-click the displayed question marks (**?????**) to enter the ST assignment expression.

Confirming with **Enter** will start an analysis and the found detected problems will be displayed in a tooltip.

Animation of the memory words used inside the ST assignment expression can be seen in a **Variable Window**. Please refer to *Variable Display Window* *(see page 1485)*.

## Memory Addresses

A memory address entered in the ST assignment expression must start with one of the following symbols, specifying the address type:

- %M (memory bit)
- %I (discrete input)
- %IW (input word)
- %MW (memory word)

The address type must be followed by an address number (no leading zeros required).

## Creating Variables Automatically

After confirming the ST assignment expression with **Enter**, for every entered address a variable will be created automatically, which is used in the equation network.

Examples

| Address | Variable |
|---|---|
| %M1 | _000001 |
| %I1 | _100001 |
| %IW1 | _300001 |
| %MW1 | _400001 |

The variable type is derived from its address suffix.

| Suffix | | Variable Type |
|---|---|---|
| no suffix | %Mx, %Ix | EBOOL |
| | %IWx, %MWx | UINT |
| I | | INT |
| DI | | DINT |
| UD | | UDINT |
| R | | REAL |
| W | | WORD |

The variable is mapped to the associated internal located memory.

For example: %IW1DI becomes the variable `_300001_DI:DINT@%IW1`.

## Constants

Constants in equation networks have the same syntax as in other IEC editors (that is without a leading # character).

## Operators

Operators and their precedence

| Operation | Symbol | Precedence |
|---|---|---|
| Parentheses | (expression) | HIGHEST |
| Function evaluation | identifier (argument list) | - |
| Negation | - | - |
| Complement | ~ | - |
| Exponentiation | ** | - |
| Multiply | * | - |
| Divide | / | - |
| Add | + | - |
| Subtract | - | - |
| Comparison | < , > , <= , >= | - |
| Equality | = | - |
| Inequality | <> | - |
| Boolean AND | & | - |
| Boolean Exclusive OR | ^ | - |

| Operation | Symbol | Precedence |
|---|---|---|
| Boolean OR | \| | - |
| Assignment | := | LOWEST |

The operators **, *, /, +, -, <<, >>, <, >, <=, >=, =, <>, &, ^ and | are supported as they are implemented now in Unity Pro ST syntax. That is not all data type combinations for operand types and the result type are supported.

**NOTE:** Unity Pro only supports a floating point value for the ** operator (EXPT-functions).

**NOTE:** Bitwise logical AND, OR or XOR operations work like in ST syntax.

## Ternary Conditional

Ternary conditional (C?t:f ), in contrast to legacy LL984, is not available in Unity Pro ST syntax.

The already existing `SEL` function block provides this functionality and has to be used instead.

Example:

**Legacy syntax:**

```
400010 := 300010 < 300020 ? 300004 : 300005
```

**Unity Pro syntax:**

```
%MW10 := SEL(%IW10 < %IW20, %IW4, %IW50);
```

With the following, automatically created by the editor:

```
_400010 := SEL(_300010 < _300020, _300004, _300005);
```

## Bitwise Shift

Bitwise shift (<<, >>), in contrast to legacy LL984, is not available in Unity Pro ST syntax.

The already existing `SHL` and `SHR` function blocks provide this functionality and have to be used instead.

Example:

**Legacy syntax:**

```
400010 = 300010U << 3
```

**Unity Pro syntax:**

```
%MW100W := SHL(%IW10W, 3);
```

With the following, automatically created by the editor:

```
_400100_W := SHL(_300010_W, 3);
```

## Function Blocks

In general you can use the function blocks provided for ST language.

In particular the following Unity Pro function blocks replace the legacy function blocks listed below.

| Legacy Function Name | Unity Pro Function Name | Argument Type* | Return Type* | Meaning |
|---|---|---|---|---|
| ABS | ABS | S,U,L,UL,F | S,U,L,UL,F | absolute value |
| ARCCOS | ACOS | F | F | arc cosine |
| ARCSIN | ASIN | F | F | arc sine |
| ARCTAN | ATAN | F | F | arc tangent |
| COS | COS | F | F | cosine |
| COSD | COSD | F | F | cosine of degrees |
| EXP | EXP | F | F | exponent function (power of e) (does not need to be a whole number) |
| FIX | REAL_TO | F | L | convert floating point to integer (presumes an FP argument) |
| FLOAT | INT_TO | S, U, L, UL | F | converts integer to floating point (presumes an integer argument) |
| LN | LN | F | F | natural logarithm (base e) |
| LOG | LOG | F | F | common logarithm (base 10) |
| SIN | SIN | F | F | sine of radians |
| SIND | SIND | F | F | sine of degrees |
| SQRT | SQRT | F | F | square root |
| TAN | TAN | F | F | tangent of radians |
| TAND | TAND | F | F | tangent of degrees |
| *: S=INT, U=UINT, L=DINT, UL=UDINT, F=REAL | | | | |

## Example of an LL984 Expression

**Legacy syntax:**

```
40701 = 40702U + COS(40703UL) * #8.00135F + SIN(40704);
```

**Unity Pro syntax:**

```
%MW701 := REAL_TO_UINT(WORD_TO_REAL(%MW702W) + COS(WORD_AS_REAL(%MW703W,
%MW704W)) * 8.00135 + SIN(WORD_TO_REAL(%MW704W)));
```

With the following, automatically created by the editor:

```
_400701 := REAL_TO_UINT(WORD_TO_REAL(_400702_W) +
COS(WORD_AS_REAL(_400703_W, _400704_W)) * 8.00135 +
SIN(WORD_TO_REAL(_400704_W)));
```

# Section 27.7
# Assigning Variables to LL984 Objects

## Assigning Variables to LL984 Objects

### Introduction

In the LL984 Editor you can assign a variable (like `_400001` or `sym1`) or an address (like %M1) to an FFB pin or to a contact/coil.

If the variable is not declared, the **Create variable?** dialog opens.

If the variable or address assigned is not compatible with the pin or contact/coil, a message will be displayed in the output window after **Build** → **Analyze**.

### Automatic Variable Naming

If you, for example, assign an address (e.g. %MW1) to a pin, the **Create variable?** dialog opens proposing a variable name (e.g. _400001) and a data type (e.g. UINT).

The following rules apply to this automatic proposal, depending on the address of the actual parameter, the pin type and dimension (n) in case of an array type:

| Address | Pin Type | Dimension (for array types) | Variable Name |
|---|---|---|---|
| %MW1 | UINT | 1 (n) | _400001 (_400001_n) |
| | INT | | _400001_I (_400001_In) |
| | WORD | | _400001_W (_400001_Wn) |
| %IW1 | UINT | 1 (n) | _300001 (_300001_n) |
| | INT | | _300001_I (_300001_In) |
| | WORD | | _300001_W (_300001_Wn) |
| %I1 | EBOOL | 1 (n) | _100001 (_100001_n) |
| %M1 | EBOOL | 1 (n) | _000001 (_000001_n) |

**NOTE:** For the most used pin types UINT and EBOOL no type suffix are appended to the proposed variable name.

### Use Case 1

Assignment to a pin of type UINT

| User-Assigned Address | Variable | Variable Created and Assigned | Variable Type |
|---|---|---|---|
| %MW1 | not existing yet | _400001 | UINT |

**NOTE:** As the pin type was UINT no type suffix was appended to the variable name.

### Use Case 2

Assignment to a pin of type ARRAY[1...n] of UINT

| User-Assigned Address | Variable | Variable Created and Assigned | Variable Type |
|---|---|---|---|
| %MW1 | not existing yet | _400001_n | ARRAY[1...n] of UINT |

### Use Case 3

Assignment to a pin of type ANY or ANY_ARRAY_UINT or ANY_ARRAY_BOOL (dynamic array)

The dimension (n) of a dynamic array will be provided by another dimension pin (e.g. LENGTH) of the function block.

This dimension pin will be initialized during block insertion.

The dimension of the array will be calculated based on the dimension pin value entered as actual parameter of the dimension pin.

- If the resulting type is a UINT array:
  dimension = dimension pin value.
- If the resulting type is a Boolean array:
  dimension = dimension pin value * 16.

| User-Assigned Address | Variable | Variable Created and Assigned | Variable Type |
|---|---|---|---|
| %MW1 | not existing yet | _400001_n | ARRAY[1...n] of UINT |

If you modify n on the dimension pin to n2:

| User-Assigned Address | Variable | Variable Created and Assigned | Variable Type |
|---|---|---|---|
| %MW1 | not existing yet | _400001_n2 | ARRAY[1...n2] of UINT |
| %MW1 | already existing | _400001_n2<br>attributes (like descriptors, comments etc.) are taken over | ARRAY[1...n2] of UINT |

### Use Case 4

Assignment to a pin of type DDT1

| User-Assigned Address | Variable | Variable Created and Assigned | Variable Type |
|---|---|---|---|
| %MW1 | not existing yet | _400001_DDT1 | DDT1 |

### Memory Consumption

If you create any kind of located variables (elementary type, array or DDT) this will not increase memory consumption on the PLC (except for Upload information).

### New Variable Creation

If you assign a different address or modify a dimension pin, a new variable will be created.

The old attributes (except the address) will be overtaken and the old variable will be deleted, if it is not used anywhere else in the program.

### Invalid Variables

If you create a located variable with an address range which exceeds the address range defined under **State RAM** of the CPU´s **Configuration** tab, the following message will be displayed after **Build** → **Analyze**:

```
Topological address index is not valid.
```

There are different ways to solve this issue:

- Click the message displayed in the output window to open the **Data Editor** and adapt the address of the variable.
- Click the message displayed in the output window to open the **Data Editor** and delete the variable and create a new variable with an applicable address range.
- Adapt the address range defined under **State RAM** of the CPU´s **Configuration** tab.

**NOTE:** If you just create a new variable at the pin of the function block without deleting the previously created, invalid variable the message mentioned above will be displayed again after next **Build** → **Analyze**.

**Example:**

Under **State RAM** of the CPU´s **Configuration** tab you entered a value 10000 for **%MW**.

In your program you have an `L_BLKM` function block with `LENGTH` = 10.

At the `SOURCE` pin of the function block you enter following value: `%MW9999`.

This results in creation of the following variable:

```
_409999:ARRAY[1..10] OF UINT@%MW9999
```

As the start address is `%MW9999` and the length is 10, the end address will be `%MW10009`.

This exceeds the address range (10000) defined under **State RAM** of the CPU´s **Configuration** tab.

Therefore the message mentioned above will be displayed after **Build** → **Analyze**.

# Section 27.8
## Calling Subroutines

## Calling Subroutines

### Introduction

Calling subroutines is managed by 3 function blocks:

- L9_JSR
- L9_LAB
- L9_RET

For detailed information please refer to the function block descriptions *(see Unity Pro, Original LL984, Block Library)*.

Up to 1023 subroutines can be established.

Unity Pro does not support interrupt handling with LL984 subroutines.

### Restrictions

For the calling subroutines feature the following restrictions apply:

- Only one LL984 segment can be created inside the subroutine (**SR Section**) node of the **MAST** task.
- L9_LAB and L9_RET function blocks can only be placed in an LL984 subroutine network in the **SR Sections**.
- An L9_LAB function block can only be placed in row 1, column 1 of an LL984 subroutine network.

### Subroutine Structure

A subroutine is identified by a number (1.. 1023).

Subroutines are called by the L9_JSR function block within an LL984 network (normal or subroutine network).

Two additional function blocks, which are only available in subroutine networks, mark the starting point (L9_LAB function block) and ending point (L9_RET function block) of a subroutine.

A subroutine may span multiple networks.

The L9_RET function block is optional and this explicit ending point of a subroutine can be omitted,

- if the last subroutine network is the last network of the subroutine segment or
- if a new subroutine network (L9_LAB) would entered.

The L9_RET function block or the implicit forms, return from the current subroutine network to the calling L9_JSR function block with an OK status.

## Nesting

A subroutine may call other subroutines or recursively call itself up to 100 times.

If the maximum nesting level is reached, an implicit return is performed and the ERR output of the L9_JSR function block is set.

A nesting level problem message is propagated to the (99) cascaded L9_JSR function blocks.

Further cascaded L9_JSR function blocks, which are being executed after a nesting level problem, do not succeed to call a subroutine and the ERR output is being set until the nesting level stack has been unwound.

**NOTE:** The message is only visible at those L9_JSR function blocks, contributing to the problem. A dedicated problem treatment is possible on each calling level. Subsequent L9_JSR function blocks of a segment are not influenced by this problem.

Please also refer to the L9_JSR function block description *(see Unity Pro, Original LL984, Block Library)*.

# Section 27.9
# Entering Comments

## Entering Comments

### Introduction

Comments can be placed in LL984 sections in the form of text objects.

Text objects can overlap other objects.

The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill further grid units.

The same rules apply for entering text and navigating within text objects as for editing ASCII text in standard text editors. (To create a line break in text objects you must press the **Ctrl+Enter** key combination).

**NOTE:** Keep in mind that each change to a comment (e.g. change to a comment text, change to the size of the text object) makes it necessary to recreate the section involved (**Build → Build Changes**).

### Selecting the Text Object

You have the following options to select text objects:

- Use the **Edit → New → Comment** menu command.
- Use the **Comment** menu from the context menu.
- Press the **F8** key.
  or
- Select the  symbol.

The active placement mode for text objects is indicated by the  cursor symbol.

## Placing Text Objects

| Step | Action |
|---|---|
| 1 | Select the text object. |
| 2 | Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>**Result:** The text object is inserted.<br>Example:<br>  |
| 3 | Enter the comment. |
| 4 | Confirm the text entered with:<br>● Clicking with the mouse outside of the text object<br>or<br>● Pressing the **Enter** key. |
| 5 | To insert further text objects:<br>● Click the target cell in the LL984 section.<br>or<br>Use the **Arrow** keys to move the gray field to the target position and press **Enter**.<br>● To return to selection mode:<br>Press the **Esc** key.<br>● To insert other objects:<br>Select the object you wish to insert. |

# Section 27.10
# Customize LL984 Editor Colors

## Customize LL984 Editor Colors

### Overview

To customize LL984 editor colors, modify the 984Colors.ini in the Unity Pro folder.

For more detail on destination folder when Unity Pro is installed, refer to the *Unity Pro Installation Manual (see Unity Pro, Installation Manual)*.

**NOTE:** Make a copy of this file before modifying it.

The LL984 editor colors are described in the table below.

### Description

This table describes the elements of the color file:

| Parameter | Default value | Description |
|---|---|---|
| `BkAnimation` | `192,192,192` | Editor background color when animation is ON and you are connected in programming mode. |
| `StatementError` | `255,0,0` | Element in error is underlined with this color. |
| `InspectBk` | `0,255,255` | Inspect window background color |
| `InspectText` | `0,0,0` | Inspect window text color |
| `InspectBkMin` | `255,255,0` | Inspect window background color when the value is less than the minimum value defined in the inspect window settings. |
| `InspectBkMax` | `255,0,255` | Inspect window background color when the value is greater than the maximum value defined in the inspect window settings. |
| `BkAnimMonitoring` | `255,211,211` | Editor background color when animation is ON and you are connected in monitoring mode. |
| `AnalyzeError` | `0,0,255` | Drawing color when an element generates an analyze error. |
| `GraphAnimBoolTrue` | `0,150,0` | Color of `ANY_BOOL` variable when its value is TRUE. |
| `GraphAnimBoolFalse` | `255,0,0` | Color of `ANY_BOOL` variable when its value is FALSE. |
| `BkComment` | `255,255,204` | Background color of comment area |
| `BkCommentEdit` | `225,225,225` | Background color of comment area when you are editing it. |
| `BkAnimNumeric` | `255,255,0` | Background color of numeric variables |
| `BlockBreakpoint` | `128,0,0` | Bullet color in the element where the breakpoint is set. |

| Parameter | Default value | Description |
|---|---|---|
| BlockCurrentStep | 255,255,0 | Color of the current element in debug mode |
| AnchorLink | 128,0,0 | Color of anchor link |
| LLFgExpression | 0,0,0 | Text color of boolean expressions |
| LLFgAddress | 100,100,230 | Text color of variable address |
| LLFgSymbol | 0,0,0 | Text color of variable name |
| LLFgComment | 0,128,0 | Text color of variable comment |
| LLBkExpression | 255,255,255 | Background color of boolean expressions |
| LLBkAddress | 255,255,255 | Background color of variable address |
| LLBkSymbol | 255,255,255 | Background color of variable name |
| LLBkComment | 255,255,255 | Background color of variable comment |
| 984CoilAlignLine | 192,192,192 | Colors of coils align line |
| 984BkEqNet | 228,228,228 | Background color of equation network |

# Chapter 28
# Programming Languages Data Selection Dialog Box

## Overview

This chapter describes the dialog boxes provided for data selection that can be called from the various programming languages.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Variable selection dialog box | 1182 |
| FFB Selection Dialog box | 1184 |
| FFB Type Selection Dialog box | 1187 |
| Data selection dialog box | 1189 |
| Function Input Assistant | 1194 |
| Data Properties | 1197 |
| Generate variables | 1199 |

# Variable selection dialog box

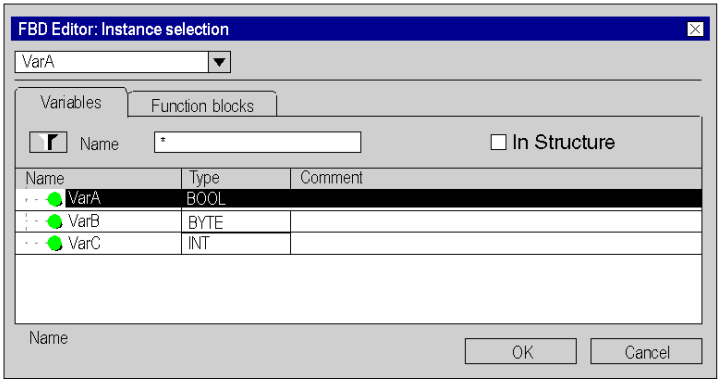
## Structure of the properties dialog box

The variable selection dialog box consists of two tabs:

- **Variables**
  You can select an elementary variable or an element of a multi-element variable in this tab.
- **Function Blocks**
  You can select a formal parameter of a function block instance in this tab.

## Variables tab

Representation of the tab **Variables** tab:

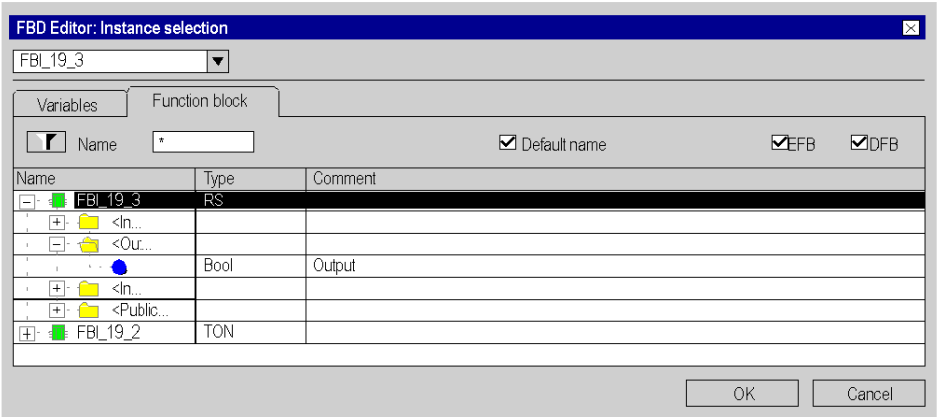



Elements of the **Variables** tab:

| Element | Description |
|---|---|
| Text box | The selected variables are displayed in this text box. |
| button  | Double click the command button to open a dialog box for filtering *(see page 412)* variables. |
| **Name** | The name of the variable to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **In Structure** | When you activate this check box, all structured variables are displayed. |
| **Name** | The names of the variables are displayed in this column. |
| **Type** | The types of the variables are displayed in this column. |
| **Comment** | The comments for the variables are displayed in this column. |

## Function Blocks tab

Representation of the **Function Blocks** tab:




Elements of the **Function Blocks** tab:

| Element | Description |
|---|---|
| Text box | The selected function block instances are displayed in this text box. |
| button  | Use this button to open a dialog box for filtering *(see page 412)* function block instances. |
| **Name** | The name of the function block instance to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **Default name** | When you activate this check box, the function block instances are displayed whose default names (e.g. FBI_19_3, TON_1) have not been changed. |
| **EFB** | When you activate this checkbox, the instances of elementary function blocks (EFBs) are displayed. |
| **DFB** | When you activate this checkbox, the instances of derived function blocks (DFBs) are displayed. |
| **Name** | This column displays the names of the function block instances and their formal parameters (inputs, outputs, public variables). |
| **Type** | This column displays the types of the function block instances and the data types of their formal parameters (inputs, outputs, public variables). |
| **Comment** | This column displays comments about the function block instances and their formal parameters (inputs, outputs, public variables). |

**NOTE:** Help for the Type..
Hold down the **Alt+F1** keys and then click on the function block type for which you wish to access Help.

# FFB Selection Dialog box

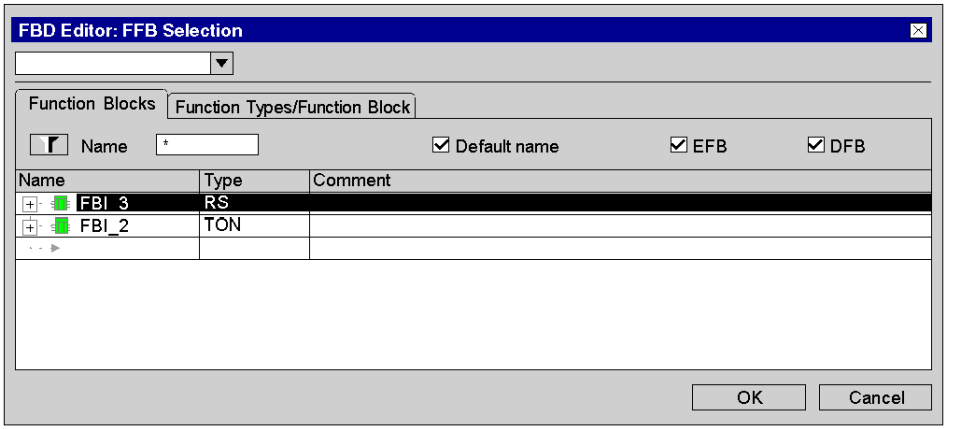
## Structure of the properties dialog box

The dialog box for selecting function block instances and function types/function block types consists of two tabs in FBD and LD:

- **Function Blocks**
  You can select a function block instance in this tab.
- **Function Types/Function Block Types**
  You can select a function type or function block type in this tab.

## Function Blocks tab

Representation of the **Function Blocks** tab:




Elements of the**Function Blocks** tab:

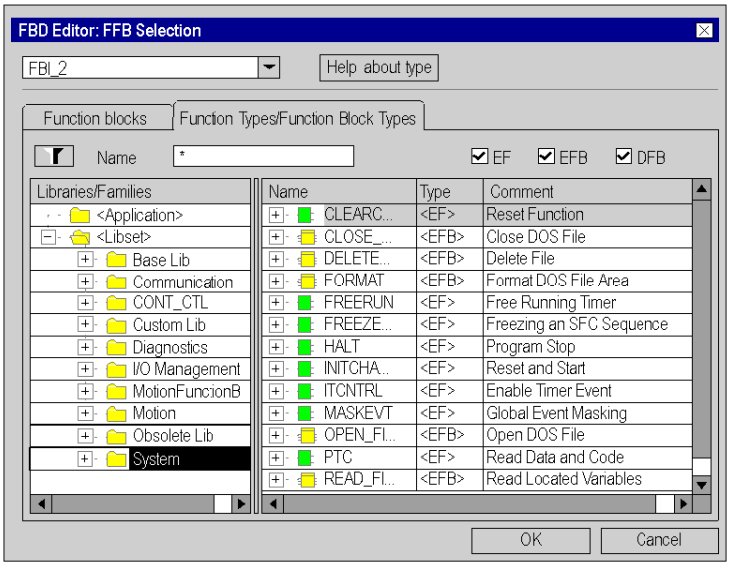
| Element | Description |
|---|---|
| Text box | The selected function block instances are displayed in this text box. |
| button  | Double click the button to open a dialog box for filtering *(see page 412)* function block instances. |
| **Name** | The name of the function block instance to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **Default name** | When you activate this check box, the function block instances are displayed whose default names (e.g. FBI_19_3, TON_1) have not been changed. |
| **EFB** | When you activate this checkbox, the instances of elementary function blocks (EFBs) are displayed. |

| Element | Description |
|---|---|
| **DFB** | When you activate this checkbox, the instances of derived function blocks (DFBs) are displayed. |
| **Name** | This column displays the names of the function block instances and their formal parameters (inputs, outputs). |
| **Type** | This column displays the types of the function block instances and the data types of their formal parameters (inputs, outputs). |
| **Comment** | This column displays comments about the function block instances and their formal parameters (inputs, outputs). |

**NOTE:** Help for the Type..
Hold down the **Alt+F1** keys and then click on the function block type for which you wish to access Help.

## Function Types/Function Block Types tab

Representation of the **Function Types/Function Block Types** tab:



Elements of the **Function Types/Function Block Types** tab:

| Element | Description |
|---|---|
| Text box | The selected function block instances are displayed in this text box. |
| **Help about type** | Use this button to open the help for the function block types shown.<br>**Note:**<br>Alternatively, you can also hold down **Alt+F1** and then click on the function block type for which you want to access Help. |
| button  | Use this command button to open a dialog box for filtering *(see page 412)* function and function block types. |
| **Name** | The name of the function type or function block type to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **EF** | When you enable this check box, Elementary Function types (EFs) are displayed. |
| **EFB** | When you enable this checkbox, Elementary Function Block types (EFBs) are displayed. |
| **DFB** | When you activate this checkbox, Derived Function Blocks (DFBs) are displayed. |
| **Libraries/Families** | The scope of the displayed function types and function block types is selected in this column:<br>● To display all function types and function block types used in the project, select **<Application>**.<br>● To display all function types and function block types of a library independently of the library assignment, select **<Libset>**.<br>● To display all function types and function block types of a library, select the desired library.<br>● To display all function types and function block types of a family, select the desired family. |
| **Name** | This column displays the names of the function types and function block types and their formal parameters (inputs, outputs). |
| **Type** | This column displays the types of the function types and function block types and the data types of their formal parameters (inputs, outputs). |
| **Comment** | This column displays comments about the function types and function block types and their formal parameters (inputs, outputs). |

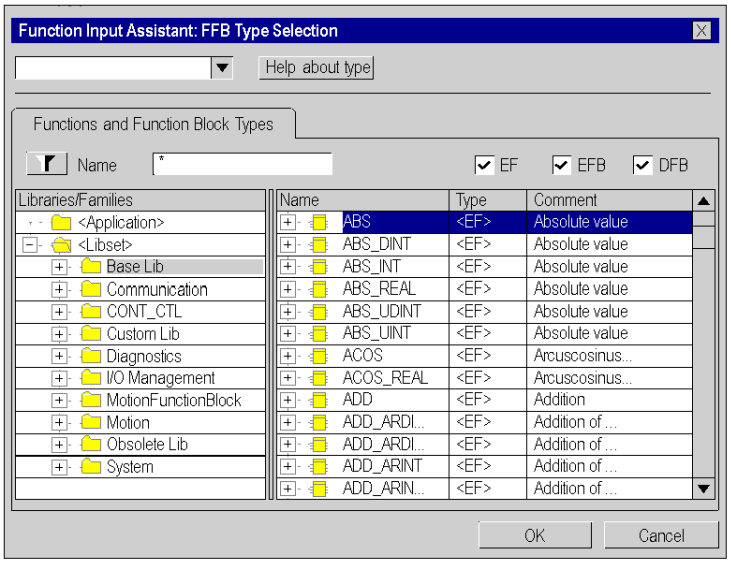
# FFB Type Selection Dialog box

## Structure of the properties dialog box

In the **Function Input Assistant dialog box: FFB type selection** you can select a function type or function block type in this tab..

## Function Input Assistant: FFB Type Selection

Representation of the dialog box:




Elements of the **Function Types/Function Block Types** dialog box:

| Element | Description |
|---|---|
| Text box | The selected function or function block types are displayed in this text box. |
| **Help about type** | Use this button to open the help for the function block types shown.<br>**Note:**<br>Alternatively, you can also hold down **Alt+F1** and then click on the function block type for which you want to access Help. |
| button<br>  | Use this command button to open a dialog box for filtering<br>*(see page 412)* function and function block types. |
| **Name** | The name of the function type or function block type to search for can be entered in this text box. You can also use the joker * and **?**. |

| Element | Description |
|---|---|
| **EF** | When you enable this check box, Elementary Function types (EFs) are displayed. |
| **EFB** | When you enable this checkbox, Elementary Function Block types (EFBs) are displayed. |
| **DFB** | When you activate this checkbox, Derived Function Blocks (DFBs) are displayed. |
| **Libraries/Families** | The scope of the displayed function types and function block types is selected in this column:<br>● To display all function types and function block types used in the project, select **<Application>**.<br>● To display all function types and function block types of a library independently of the library assignment, select **<Libset>**.<br>● To display all function types and function block types of a library, select the desired library.<br>● To display all function types and function block types of a family, select the desired family. |
| **Name** | This column displays the names of the function types and function block types and their formal parameters (inputs, outputs). |
| **Type** | This column displays the types of the function types and function block types and the data types of their formal parameters (inputs, outputs). |
| **Comment** | This column displays comments about the function types and function block types and their formal parameters (inputs, outputs). |

# Data selection dialog box

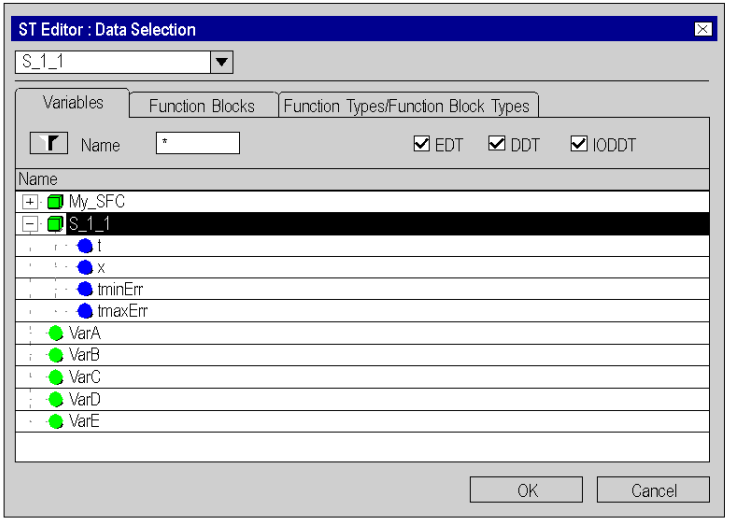
## Structure of the properties dialog box

The data selection dialog box consists of three tabs in IL and ST:

- **Variables**
  You can select an elementary variable or an element of a multi-element variable in this tab.
- **Function Blocks**
  You can select a function block instance in this tab.
- **Function Types/Function Block Types**
  You can select a function type or function block type in this tab.

## Variables tab

Representation of the tab **Variables**tab:

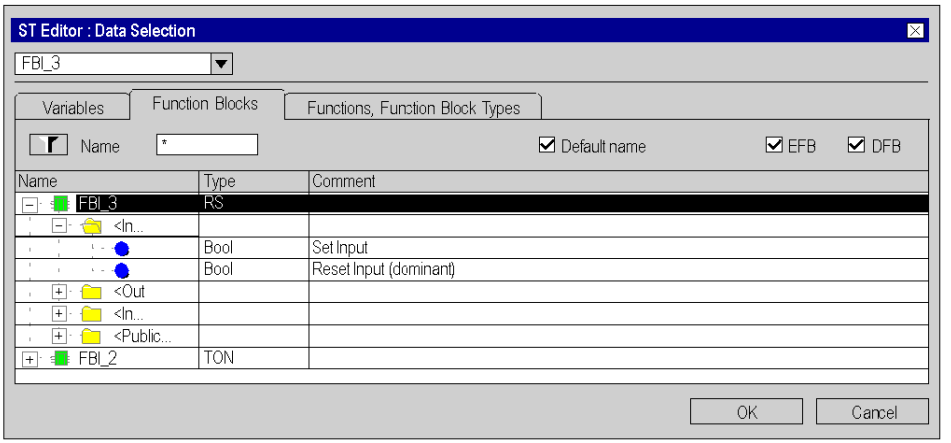


Elements of the **Variables** tab:

| Element | Description |
|---|---|
| Text box | The selected variables are displayed in this text box. |
| button  | Double click the button to open a dialog box for filtering *(see page 412)* variables. |
| **Name** | The name of the variable to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **EDT** | When you activate this check box, elementary variables are displayed. |
| **DDT** | When you activate this checkbox, multi-element variables (derived variables) are displayed. |
| **IODDT** | When you activate this checkbox, derived I/O variables (IODDT variables) are displayed. |
| **Name** | The names of the variables are displayed in this column. |

## Function Blocks tab

Representation of the **Function Blocks** tab:



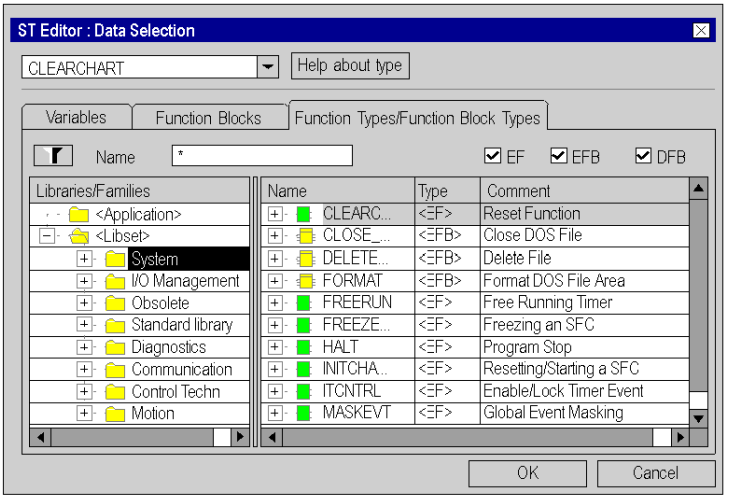
Elements of the**Function Blocks** tab:

| Element | Description |
|---|---|
| Text box | The selected function block instances are displayed in this text box. |
| button  | Use this button to open a dialog box for filtering *(see page 412)* function block instances. |
| **Name** | The name of the function block instance to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **Default name** | When you activate this check box, the function block instances are displayed whose default names (e.g. FBI_3, TON_1) have not been changed. |
| **EFB** | When you activate this checkbox, the instances of elementary function blocks (EFBs) are displayed. |
| **DFB** | When you activate this checkbox, the instances of derived function blocks (DFBs) are displayed. |
| **Name** | This column displays the names of the function block instances and their formal parameters (inputs, outputs, public variables). |
| **Type** | This column displays the types of the function block instances and the data types of their formal parameters (inputs, outputs, public variables). |
| **Comment** | This column displays comments about the function block instances and their formal parameters (inputs, outputs, public variables). |

**NOTE:** Help for the Type..
Hold down the **Alt+F1** keys and then click on the function block type for which you wish to access Help.

## Function Types/Function Block Types tab

Representation of the **Function Types/Function Block Types** tab:




Elements of the **Function Types/Function Block Types** tab:

| Element | Description |
|---|---|
| Text box | The selected function block types are displayed in this text box. |
| **Help about type** | Use this button to open the help for the function block types shown.<br>**Note:**<br>Alternatively, you can also hold down **Alt+F1** and then click on the function block type for which you want to access Help. |
| button | Use this command button to open a dialog box for filtering *(see page 412)* function and function block types. |
| **Name** | The name of the function type or function block type to search for can be entered in this text box. You can also use the joker * and **?**. |
| **EF** | When you enable this check box, Elementary Function types (EFs) are displayed. |
| **EFB** | When you enable this checkbox, Elementary Function Block types (EFBs) are displayed. |
| **DFB** | When you activate this checkbox, Derived Function Blocks (DFBs) are displayed. |

| Element | Description |
|---|---|
| **Libraries/Families** | The scope of the displayed function types and function block types is selected in this column:<br>● To display all function types and function block types used in the project, select **<Application>**.<br>● To display all function types and function block types of a library independently of the library assignment, select **<Libset>**.<br>● To display all function types and function block types of a library, select the desired library.<br>● To display all function types and function block types of a family, select the desired family. |
| **Name** | This column displays the names of the function types and function block types and their formal parameters (inputs, outputs). |
| **Type** | This column displays the types of the function types and function block types and the data types of their formal parameters (inputs, outputs). |
| **Comment** | This column displays comments about the function types and function block types and their formal parameters (inputs, outputs). |

# Function Input Assistant

## Introduction

If **no**FFB is selected when the function input assistant is called the FFB input assistant is used for assigning parameters to the new FFBs generated.

Example for:

- FBD *(see page 633)*
- LD *(see page 779)*
- IL *(see page 1057)*
- ST *(see page 1089)*

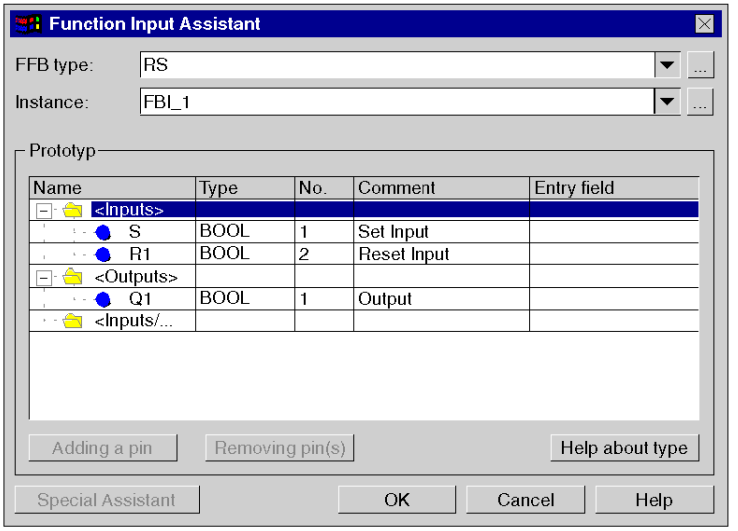
If an FFB is selected when the function input assistant is called the FFB input assistant is used for assigning parameters to the new FFB generated.

Example for:

- FBD *(see page 652)*
- LD *(see page 799)*
- IL *(see page 1067)*
- ST *(see page 1098)*

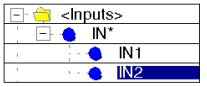
## Function Input Assistant

Representation:



Elements:

| Element | Description |
|---|---|
| **FFB type** | Enter the function type or function block type to edit in this text box.<br>You have the following options to enter the function type or function block type:<br>● You can directly enter the type.<br>● You can select the type from the list of recently used types using the ▾ symbol.<br>or<br>● Using the button **...** you can open a dialog box to select FFB *(see page 1185)* types. |
| **Instance** | For existing function blocks this textbox shows the name of the selected function block instance.<br>For function blocks to be generated enter the name of the selected function block instance in this textbox.<br>You have the following options to enter the function instance or function block instance:<br>● You can directly enter the instance.<br>● You can select the instance from the list of recently used instances using the ▾ symbol.<br>or<br>● Using the button **...** a dialog box for FFB instances selection *(see page 1184)* . |
| **Name** | This column displays the formal parameters (inputs, outputs) of the function instance or function block instance. |
| **Type** | This column displays the data types of the formal parameters (inputs, outputs) of the function instance or function block instance. |
| **No.** | This column displays the pin positions of the formal parameters (inputs, outputs) of the function instance or function block instance. |
| **Comment** | This column displays the comments of the formal parameters (inputs, outputs) of the function instance or function block instance. |
| **Entry field** | Enter the actual parameters of the function instance or function block instance in this column.<br>You have the following options to enter actual parameters:<br>● You can directly enter the actual parameter.<br>● You can select the actual parameter from the list of recently used actual parameters using the ▾ symbol.<br>or<br>● Using the button **...** a dialog box for variables selection *(see page 1182)* . |

| Element | Description |
|---|---|
| **Adding a pin** | Using this button you can increase thenumber of inputs *(see page 656)* for expandable functions (e.g. AND, OR). Expandable pins can be recognized from the following display:<br>  <br><br>To add further pins select the last pin in the structure and confirm using the **Add pin** button. |
| **Removing pin(s)** | Using this button you can decrease the number of inputs *(see page 656)* for expandable functions (e.g. AND, OR). To decrease the number of inputs select the last pin(s) in the structure and use the **Remove pin(s)** button. |
| **Help about type** | Use this button to call up the FFB. |
| **Special Assistant** | Some complex FFBs provide additional windows for entering actual parameters. To access these windows use the **Special Assistant** button.<br>**NOTE:** take care that for the same FFB the **Special Assistant** could be available for some platforms and not for other platforms. |
| **OK** | For new FFBs the FFB insert mode for placing FFBs becomes active. Use this button to accept the allocated actual parameters for existing FFBs. |
| **Close** | With this button, you can close the dialog box without accepting changes. |
| **Help** | Use this button to call up the dialog. |

# Data Properties

## Introduction

The anchored window *(see page 1889)* **Data Properties** is used to display and modify the variables and FFB properties.

The **Data Properties** window can be called up

- using the Data Editor *(see page 325)*.
  **a.** Select one or several lines in the Data Editor.
  **b.** Select **Properties** from the context menu.
- the programming language sections
  **a.** Select one or several element(s) in the programming language section.
  **b.** Select **Data Properties** from the context menu or press **CTRL+Enter**.
- the tab **I/O Objects** of a module
  **a.** Select one or several lines in the **Address/Name/Type** table.
  **b.** Select **Data Properties** from the context menu.

How the window behaves and the options in the **Data Properties** window depend on whether the window was opened using the Data Editor or the Programming Languages section.

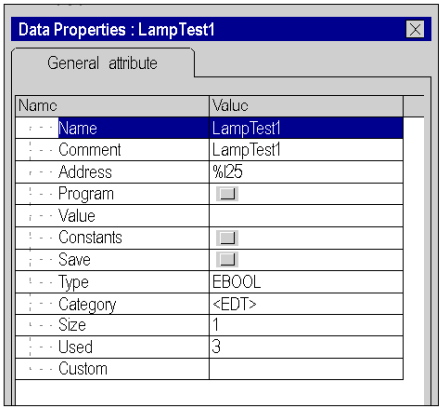
**NOTE:** If more than one object is selected, only the common data for all the selected objects is displayed. If this data is modified, the modification applies to all the selected objects. In this way it is possible to assign the same comment to different variables or assign the same initial value, see also: *Changing the attribute in several instances, page 400*

| ⚠ CAUTION |
| --- |
| **LOSS OF DATA**<br>Before changing a data in the **Data Properties** window, ensure that the change is appropriate not only to the selected instance of this data, but also to all other instances of this data.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

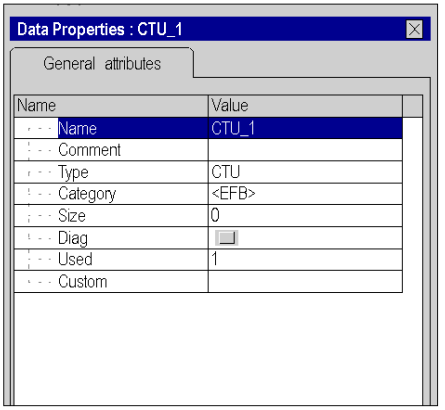
## Structure of the Window

The contents of the window depend on the selected object:

Window **Data Properties** for elementary variables:




Window **Data Properties** for function blocks:



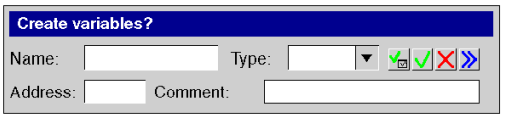

## Available Attributes

The list of available attributes is the same as the **Data Editor** columns. Refer to the Data Editor columns *(see page 404)* for more information.

# Generate variables

## Representation

Representation of the dialog box:




## Items

Elements of the dialog box:

| Element | Description |
|---|---|
| **Name** | Name of the variables to generate (maximum length = 32 characters). |
| **Type** | Datatype of the variables to generate.<br>If you want to allocate the variable for the elementary data type, you can select it from the list.<br>If you want to allocate the variables for the derived data type, enter the data type in the text box. |
|  | This button is only available in the IL and ST editor under the following conditions:<br>● The cursor is located where a variable can be inserted (i.e. not in a word).<br>● The name of the variable is automatically entered in the text box **Name**.<br>You can confirm the entries using this button and the variable will be declared and inserted in the section. |
|  | In the FBD and LD-Editor:<br>You can confirm the entries using this button and the variable will be declared and (if not already inserted) inserted in the section.<br>In the IL and ST Editor:<br>You can confirm the entries using this button and the variable will be declared (but not inserted in the section). |
|  | Cancel the generate of the variables using this button. |
|  | Use this button to show or hide the **Address** and **Commentary**text boxes. |
| **Address** | Address of the variables to generate (optional). |
| **Comment** | Comment for the variables to generate (optional). |

# Chapter 29
# I/O Objects tab

## Subject of this Chapter

This chapter describes the **I/O objects** tab, which is used to associate variables with the module I/Os and to manage these different variables.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Introduction to the I/O Object Tab | 1202 |
| I/O Objects Tab for a Processor | 1204 |
| I/O Objects Tab for a Module | 1207 |
| Details of Areas in the I/O Objects Tab | 1209 |
| I/O Objects Tab for a Bus Device | 1214 |
| How to Use the I/O Objects Tab | 1215 |

# Introduction to the I/O Object Tab

## General Points

This tab is used to manage the I/O objects of a module, a field bus device or the memory and system objects of the PLC.

The following table summarizes the different functionalities according to the type of device associated with the tab:

| Function | Processor | Module or bus device |
|---|---|---|
| Displaying all located objects of the PLC | X | - |
| Displaying I/O objects (topological addressing) | X (1) | X (2) |
| Filtering all located objects of the PLC | X | - |
| Filtering I/O objects | X (1) | X (2) |
| Sorting all located objects of the PLC | X | - |
| Sorting I/O objects | X (1) | X (2) |
| Creating an IODDT variable for a module | - | X |
| Creating an IODDT variable for a channel | - | X |
| Creating a topological EDT variable | - | X |
| Presymbolizing a set of IODDT variables for a module | - | X |
| Presymbolizing a set of topological EDT variables | - | X |
| For Quantum and Modicon M340 PLCs, displaying the correspondence between the topological address and State RAM. | X | X |
| Legend:<br>**X**: available functionality,<br>**-**: non-available functionality<br>**(1)**: of all I/O objects of the PLC<br>**(2)**: only the I/O objects of the module or the device concerned. | | |

## Access Rules

To access this tab, the following conditions must be met:

- The tab can be accessed from the configuration screen for the module, processor or bus device (it is located at the same level as the description tab).
- The tab is accessible in offline and online mode.

## Operating Rules

The main operating rules are as follows:

- This tab can be used to create and associate variables, but not to delete them.
- To delete a variable created by this tab, you must use the data editor.
- Multiple presymbolization can only be used on compatible objects (of the same type).
- For the processors, the tab only allows PLC objects to be displayed (variables cannot be associated).
- The control loops built into the processor are regarded as module channels, and consequently, creation and presymbolization are possible.
- It is possible to name a variable derived after symbolization.
  For example, if `Counter_0` is a `T_COUNT_STD` type variable, the presymbolized counting value `Counter_0.CUR_MEASURE` can be renamed (`Value_C0`, for example).

# I/O Objects Tab for a Processor

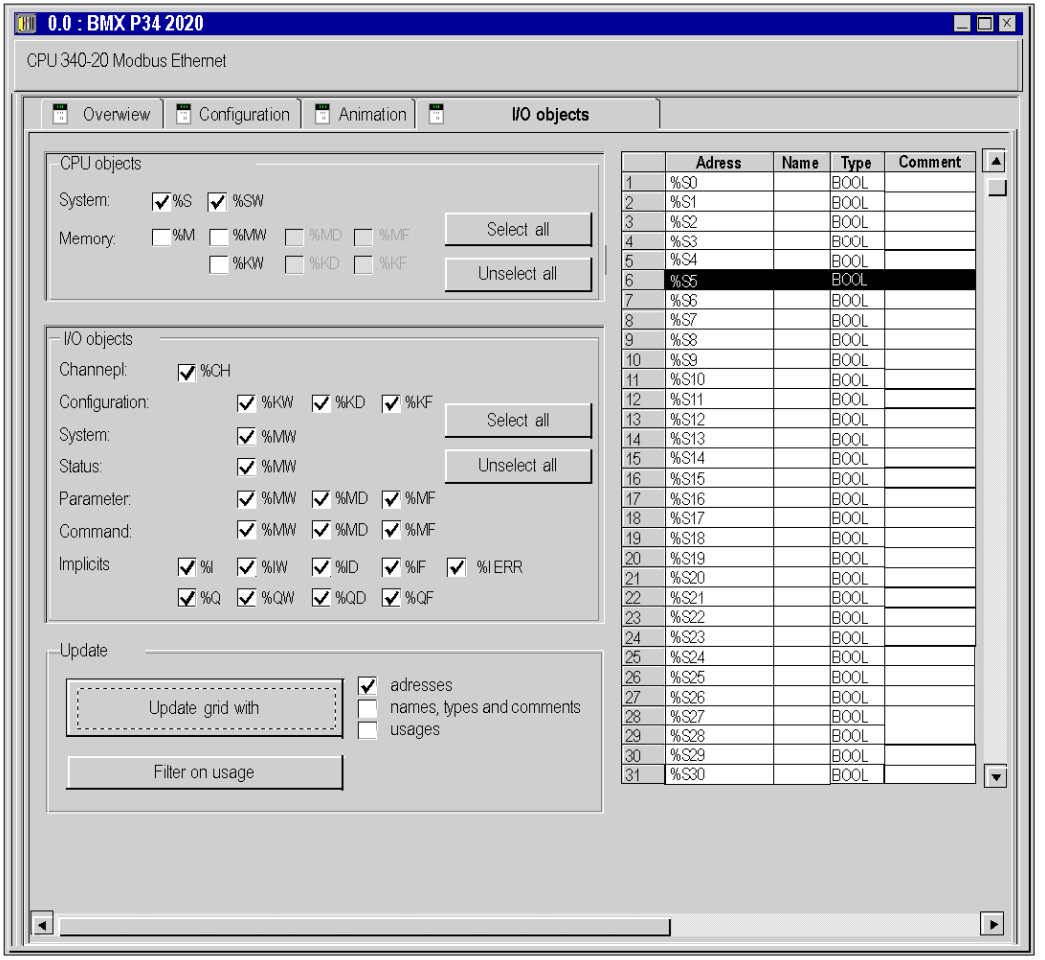
## At a Glance

The processor I/O Objects tab can be used to view PLC objects (system objects, memory objects) as well as I/O objects, regardless of whether they belong to an in-rack module or to a communication bus device (for example, Fipio).

**NOTE:** This screen can only be used to view objects; for modifications, use the **I/O objects** tab associated with the modules or bus devices *(see page 1207)*.

## Illustration

Example of an I/O Objects screen for a Modicon M340 processor.



## Description

The different areas of the screen are as follows:

| Zone | Description |
|---|---|
| CPU objects *(see page 1209)* | This zone can be used to select PLC system and memory objects that you would like to view in the **Address Name Type Comment** zone. |
| I/O Objects *(see page 1210)* | This zone can be used to select processor I/O objects that you would like to view in the **Address Name Type Comment** zone. |
| Update *(see page 1213)* | This zone allows you to start the update of the information in the **Address Name Type Comment** zone. |
| Address Name Type Comment *(see page 1212)* | Zone used to view and select data. The contents of this zone are obtained using the buttons and selections of the previous zones. The Quantum **State RAM** column is added. |

# I/O Objects Tab for a Module

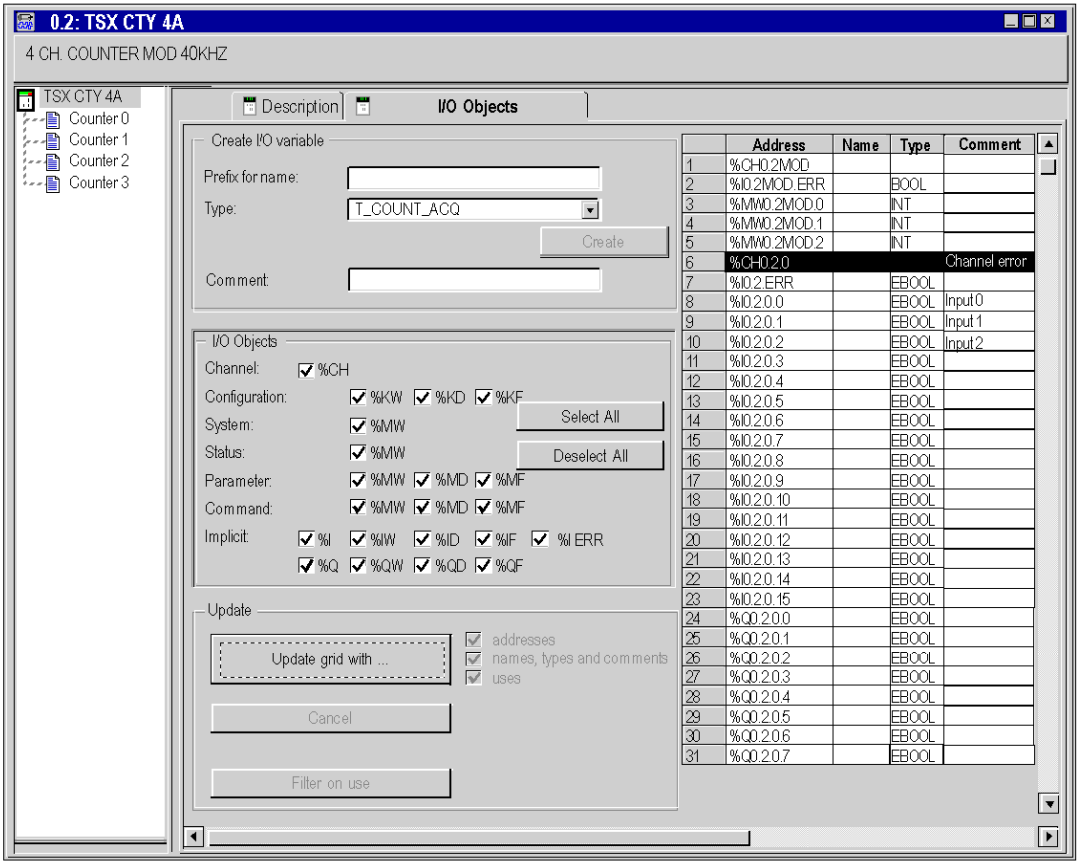
## At a Glance

For a module, this tab can be used to view module I/O objects and to manage the association of these objects with variables.

**NOTE:** To view PLC objects (system, internal memory or internal constant objects), use the **I/O objects** tab associated with the processor *(see page 1204)*.

## Illustration

Example of an I/O objects screen for an Premium X bus module.



## Description

The different areas of the screen are as follows:

| Zone | Description |
|---|---|
| Create I/O variable *(see page 1211)* | This zone can be used to create variables and to associate them with channels or channel elements with a comment for each variable. |
| I/O Objects *(see page 1210)* | This zone can be used to select processor I/O objects that you would like to view in the **Address Name Type Comment** zone. |
| Update *(see page 1213)* | This zone allows you to start the update of the information in the **Address Name Type Comment** zone. |
| Address Name Type Comment *(see page 1212)* | Zone used to view and select data. The contents of this zone are obtained using the buttons and selections of the previous zones. For Quantum and Modicon M340, the **State RAM** column is added. |

**NOTE:** For Modicon M340 I/O modules only the topological I/O objects are displayed. It is not possible to instantiate an IODDT.

# Details of Areas in the I/O Objects Tab

## At a Glance

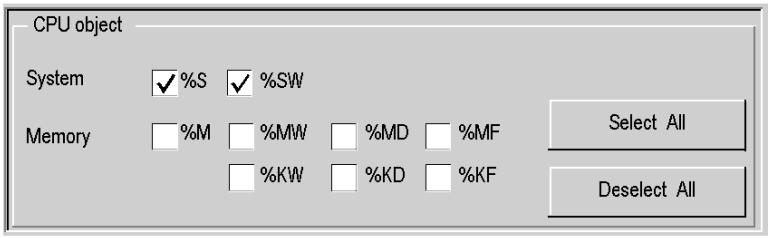
The I/O Objects tab is available for:

- Modules in the rack and their channels,
- Processors and their channels,
- Process control loops,
- Communication bus devices.

The following paragraphs give details of the areas available from this tab.

## Illustration of the CPU Objects Area

The CPU Objects area is only available for the processor. This appears as follows:




## Description of the CPU Objects Area

The selection of different objects using checkboxes %S, %SW, %M, %MW, %MD, %MF, %KW, %KD and %KF enables these to be displayed in the **Address Name Type Comment** area for Premium or the **Address Name Type State RAM Comment** area for Modicon M340 and Quantum, once the **Update Grid with...** button is pressed, in the **Update** area.

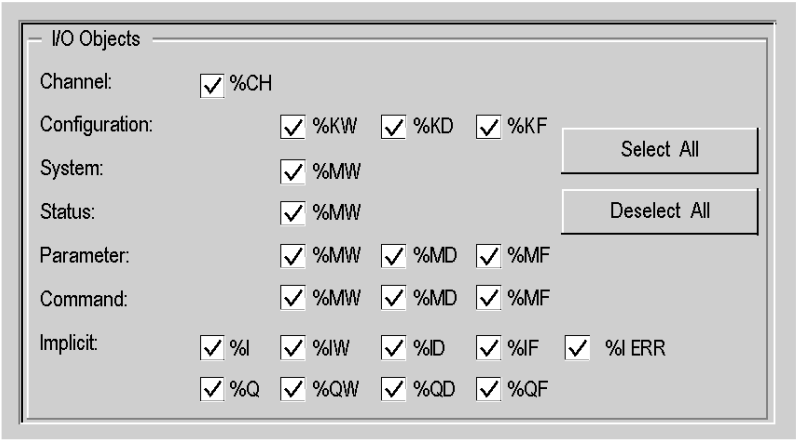
**NOTE:** For Premium/Atrium PLCs double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) should be located by an integer type (%MW<i>, %KW<i>). Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

**NOTE:** For Modicon M340 PLCs, double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) are not available.

**NOTE:** For Modicon M340 PLCs, the index (i) value must be even *(see Unity Pro, Program Languages and Structure, Reference Manual )* for double-type instances of located data (%MW and %KW).

## Illustration of the I/O Objects Area

The I/O Objects area is available for modules, processors, control loops and bus devices. It appears as follows:




## Description of the I/O Objects Area

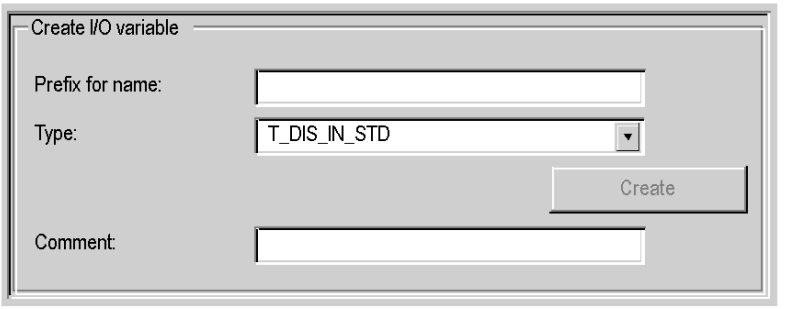
The selection of different objects using checkboxes enables these to be displayed in the **Address Name Type Comment** area, once the **Update Grid with...** button is pressed, in the **Update** area.

The different objects can be selected by type:

- **Channel**: for module channels or a bus device.
- **Configuration**: for configuration language objects.
- **System**: for language objects managing explicit exchanges.
- **State**: for status language objects (accessible by READ_STS).
- **Parameter**: for configuration language objects (accessible by READ_PARAM, WRITE_PARAM, SAVE_PARAM, RESTORE_PARAM).
- **Command**: for command language objects (accessible by WRITE_CMD).
- **Implicit**: for implicit language objects of the module or a bus device.

## Illustration of the Create I/O Variable Area

The **Create I/O Variable** area is only available for modules, control loops and bus devices. It appears as follows:




## Description of the Create I/O Variable Area

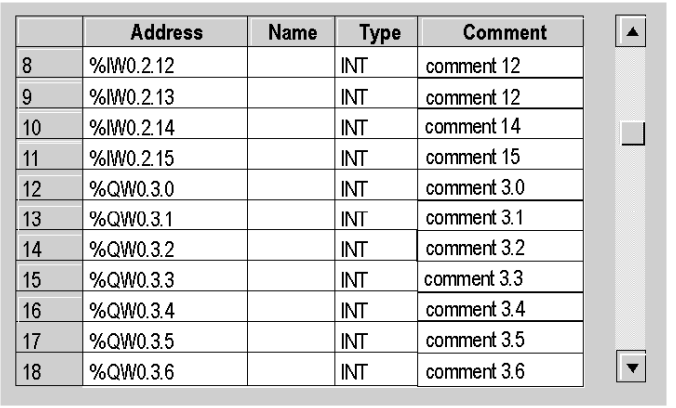
Having selected one or more objects in the **Address Name Type Comment** area, you can select an IODDT type and create one or more variables of this type by clicking **The I/O Objects tab is available for:Create**.

Operating rules:

- By selecting a line in the **Address Name Type Comment** area, you can create a variable for which you can enter the name. You can also assign a comment to the variable.
- By selecting several homogenous (same type) lines from the **Address Name Type Comment** area, you can automatically create several variables with identical prefixes (the first variable with the suffix 0 the second 1, the third 2, etc.). This method also applies for the variable comment (the first comment will have the suffix 0, the second 1, and the third 2, etc.).
- When the selected variable(s) is/are of EDT type, the type area is grayed out. Type selection is only available when several types are available.

## Illustration of the Address Name Type Comment Area

The Address Name Type Comment area is available for all I/O Objects tabs. These appear as follows:



| | Address | Name | Type | Comment |
|---|---|---|---|---|
| 8 | %IW0.2.12 | | INT | comment 12 |
| 9 | %IW0.2.13 | | INT | comment 12 |
| 10 | %IW0.2.14 | | INT | comment 14 |
| 11 | %IW0.2.15 | | INT | comment 15 |
| 12 | %QW0.3.0 | | INT | comment 3.0 |
| 13 | %QW0.3.1 | | INT | comment 3.1 |
| 14 | %QW0.3.2 | | INT | comment 3.2 |
| 15 | %QW0.3.3 | | INT | comment 3.3 |
| 16 | %QW0.3.4 | | INT | comment 3.4 |
| 17 | %QW0.3.5 | | INT | comment 3.5 |
| 18 | %QW0.3.6 | | INT | comment 3.6 |

## Description of the Address Name Type Comment Area

This area enables you to:

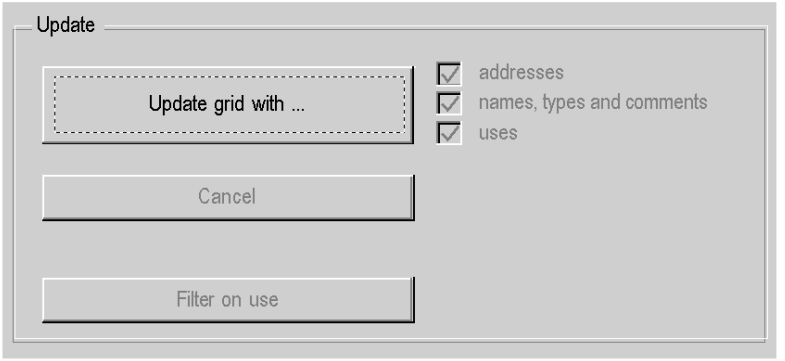
- Display the objects selected in the **CPU Objects** and **I/O Objects** areas,
- Select one or more object lines in order to create variables and associate variables with these,
- Open *(see page 1217)* the **Data Properties** window,
- View the comment associated with the variable.

For Quantum PLCs, an additional column is available:

- **State RAM** is used to view the correspondence between the topological address and State RAM.

## Illustration of the Update Area

The Update area is available for all **I/O Objects** tabs. These appear as follows:




## Description of the Update Area

By clicking the **Update Grid with...** button, you can update the **Address Name Type Comment** area from the information selected in the **CPU Objects** and **I/O Objects** areas.

The **Address Name Type Comment** area is also used to select objects in order to create variable names and comments for them (only true for modules, control loops and communication bus devices).

The **Cancel** button is used to cancel the update of object names from the **Address Name Type Comment** area.

The **Filter on use** button is used to display only those objects used in the project.

The **addresses**, **names, types and comments** and **use** checkboxes are used, respectively, to display:

- The **Address** column,
- the **Name** and **Type** and **Comments** columns,
- In bold the variables used in the program.

# I/O Objects Tab for a Bus Device

## At a Glance

All bus devices that have a description window also have an **I/O objects** tab.

Currently, **Fipio** (Premium) and **CANopen** (Modicon M340 and Modicon M580) bus devices have an **I/O objects** tab. This tab behaves in exactly the same way as the I/O objects of a module *(see page 1207)* tab.

# How to Use the I/O Objects Tab

## At a Glance

The I/O tab is used to facilitate the management of located variables in an automation project using Unity Pro.

For information on the different possible actions, refer to the pages describesing this tab for:

- Processors *(see page 1204)*
- Modules *(see page 1207)*
- Communication bus devices *(see page 1214)*

A description of the main procedures for carrying out these actions is given below.

## How to Display Required Objects

The following table shows the procedure for displaying a list of objects associated with a processor, module or communication bus device.

| Step | Action |
|---|---|
| 1 | In the **CPU objects** *(see page 1209)* area or **I/O objects** *(see page 1210)* area, check the boxes of the objects you want to display (%S, %SW, %MW, %CH, %M, %MW, %I, %Q, etc.).<br>**Note**: The **Select All** and **Deselect All** buttons can be used to check or un-check all boxes. |
| 2 | In the **Update** *(see page 1213)* area, click **Update Grid with...**.<br>**Result**: the **Address Name Type Comment** *(see page 1212)* area is updated and displays all the selected variables available on the processor, module or current bus device.<br>**Note**: When searching for names, it is possible to cancel an update in progress by clicking **Cancel**. |

## How to Select an Object

The following table shows the procedure for selecting an object associated with a processor, module or communication bus device.

| Step | Action |
|---|---|
| 1 | In the **Address Name Type Comment** *(see page 1212)* area, click the line of the object to select.<br>**Result**: the line appears in reverse video, and the **Create I/O variable** *(see page 1211)* area becomes accessible for modules and communication bus devices, for creating a variable associated with the selected line.<br>**Note**: If the **Address Name Type Comment** area is empty or contains no object to select, see the section **How to Display Required Objects** *(see page 1215)*. |

## How to Create a Variable for an Object

The following table shows the procedure for creating a variable for a module object or communication bus device object.

| Step | Action |
|---|---|
| 1 | In the **Address Name Type Comment** *(see page 1212)* area, click the line of the object to select.<br>**Result**: the line appears in reverse video, and the **Create I/O variable** *(see page 1211)* area becomes accessible for modules and communication bus devices, for creating a variable associated with the selected line.<br>**Note**: If the **Address Name Type Comment** area is empty or contains no object to select, see the section **How to Display Required Objects** *(see page 1215)*. |
| 2 | In the **Create I/O variable** *(see page 1211)* area, select the type of variable to create (if available).<br>**Note**: When the variable is an EDT type (such as DINT, INT, BOOL, EBOOL type variables) or when the available IODDT is unique, the choice of type is not available. |
| 3 | In the **Create I/O variable** *(see page 1211)* area, enter the name of the variable to be created. |
| 4 | In the **Create I/O variable** *(see page 1211)* area, enter the comment for the variable to be created. |
| 5 | In the **Create I/O variable** *(see page 1211)* area, click **Create**.<br>**Result**: a new variable is created, which is associated with the object selected in the **Address Name Type Comment** *(see page 1212)* area. |

## How to Select a List of Objects

The following table shows the procedure for selecting several objects associated with a module or communication bus device.

| Step | Action |
|---|---|
| 1 | In the **Address Name Type Comment** *(see page 1212)* area, click the left column (gray column of object numbers) on the first object to select. |
| 2 | Hold down the mouse button and slide the cursor down to the last element to select.<br>**Result**: the list of selected objects appears in reverse video. |

## How to Create Variables Associated with a List of Objects

The following table shows the procedure to create variables associated with several homogenous objects from a module or communication bus device.

| Step | Action |
|---|---|
| 1 | In the **Address Name Type Comment** *(see page 1212)* area, click the left column (gray column of object numbers) on the first object to select. |
| 2 | Hold down the mouse button and slide the cursor down to the last element to select.<br>**Note**: If the **Address Name Type Comment** area is empty or contains no objects to select, see the section **How to Display Required Objects** *(see page 1215)* in order to select a contiguous list of objects to select. |
| 3 | In the **Create I/O variable** *(see page 1211)* area, select the type of variable to create (if available).<br>**Note**: When the selected variables are of EDT type (such as DINT, INT, BOOL, EBOOL type variables) or when a single IODDT is available, the choice of type is not available.<br>**Caution**: if heterogeneous (not of the same type) objects appear in the list, the creation of variables will not be available. You must therefore select a list of homogenous objects. |
| 4 | In the **Create I/O variable** *(see page 1211)* area, enter the prefix to be associated with all selected objects.<br>**Note**: This prefix will be associated with a number to form the name of each variable. The first object will be number 0, the second 1, the third 2, etc.<br>**Example**: if you enter the prefix **Valve**, the created variables will be: Valve0, Valve1, Valve2, Valve3, etc. |
| 5 | In the **Create I/O variable** *(see page 1211)* area, enter the prefix of the comment to be associated with all selected objects.<br>**Note**: This prefix will be associated with a number to form the comment for each variable. The first object will be number 0, the second 1, the third 2, etc.<br>**Example**: if you enter the prefix **Comment**, the created comments will be: Comment0, Comment1, Comment2, Comment3, etc. |
| 6 | In the **Create I/O variable** *(see page 1211)* area, click **Create**.<br>**Result**: the new variables are created and appear in the **Address Name Type Comment** *(see page 1212)* area. |

## How to Access Data Properties

The following table provides a step-by-step procedure for accessing the **Data Properties** window from the I/O tab.

| Step | Action |
|---|---|
| 1 | In the **Address Name Type Comment** *(see page 1212)* area click on the variable whose properties you wish to display. |
| 2 | Right-click on **Properties**.<br>**Result**: the **Data Properties** window is displayed.<br>Once it is open, this window displays the properties of the variable selected in the **Address Name Type Comment** area. |

## Frequently Asked Questions

Below is a list of the most commonly asked questions you may have on the use of the I/O Objects tab.

| Question | Answer |
|---|---|
| How can I sort the objects list into alphabetical order in a column? | Simply click on the name of the column to be sorted. |
| How can I display the variables and objects used in my project? | The variables and objects used in a program appear in bold and the others are not yet used. |
| How can I filter the objects in the **Address Name Type Comment** area? | Use the check boxes of the **Update** area. |
| Why are the **Address Name Type Comment** or **Address Name Type State RAM Comment** areas empty? | Possible reasons include:<br>● You have not yet updated the area by clicking **Update Grid with...**.<br>● You have not yet checked the boxes in the **CPU Objects** or **I/O Objects** areas.<br>● No object for which the boxes are checked appears in this module, processor or communication bus device. |
| Why is the Quantum **State RAM** column empty? | Possible reasons include:<br>● The **Address Name Type State Ram Comment** area is empty, refer to the previous question.<br>● The elements displayed are not associated with State RAM. |
| Why is variable creation not available? | Possible reasons include:<br>● You are in the processor, and it is therefore impossible to display the objects.<br>● You have selected a list of heterogeneous objects, and it is only therefore possible to create lists of variables for objects of the same type.<br>● You have selected an object that is already associated with a variable. You must go into the data editor to delete this variable, in order to create another.<br>● You have selected a list of objects of which at least one is already associated with a variable (see previous point). |

# Chapter 30
# Memory Tab

## Memory Tab

### Overview

The **Memory** tab is only available for the Modicon M340 hardware platform equipped with a State RAM memory area.

It allows defining the type of memory management for the selected input / output module.

The option you select here must comply with the type of memory management you selected for the processor *(see page 1352)*.

The following settings are possible:

| If... | Then... |
|---|---|
| **Topological** memory management is configured for the processor | you can only select **Topological** memory management for the discrete and analog modules. |
| **Mixed topological and State RAM** memory management is configured for the processor | you can select either **Topological** or **Mixed topological and State RAM** memory management for the discrete and analog modules. |

The memory management settings of the modules are compared with the memory management settings of the processor during **Analyze**. If any inconsistencies are detected, an appropriate message is displayed in the output window.

The type of memory management you selected for the processor will be used as default setting for the **Memory Management** parameter in the **Memory** tab.

Any application made with Unity Pro version 6.1 and later is automatically configured with address configuration for topological addressing as the default setting.

**NOTE:** If you want to import a legacy LL984 Compact application which uses Modbus request to communicate with an HMI, you have to use State RAM addressing to preserve the Modbus exchange between PLC and HMI.

## Specific Configuration Dialogs in Case of LL984 Import

If an ST section has been created during import to emulate Compact configuration behavior *(see LL984 Editor, Reference Manual, LL984 Specifics)*, the converter creates a specific I/O module configuration with specific configuration dialogs *(see LL984 Editor, Reference Manual, LL984 Specifics)*.

The **Memory Management** area is grayed and the choice is fixed to **Mixed topological and State RAM**.

In the **State RAM** area **Format** and **Coding** are defined by the converter.

You can only delete these modules. Copy and paste of these modules is not possible.

## Memory Management

The area **Memory Management** of the **Memory** tab provides the following options:

| Option | Description |
|---|---|
| **Topological** | Select this option to work with pure topological addressing.<br>If this mode is selected, the parameters in the **State RAM** area of this dialog box are disabled. |
| **Mixed topological and State RAM** | Select this option to work with State RAM addressing for IO access.<br>Topological addressing remains possible for some IO objects (%CH, %I.ERR, ...) but it's not possible to access IO values (%I, %Q).<br>With this option selected, define further State RAM options in the **State RAM** area of this tab for the different types of modules.<br>**NOTE:** With this type of memory management, all channels are associated with only one task (MAST or FAST). |

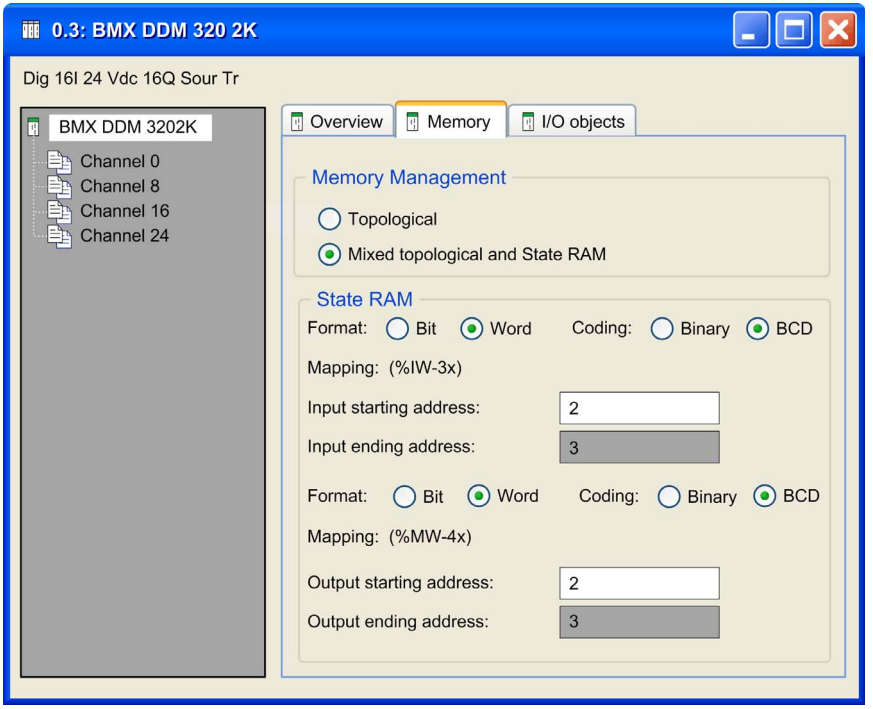
**NOTE:**
If you select **Mixed topological and State RAM** for **Memory Management** the following online features are disabled:

- I/O Objects
- Memory
- Debug
- Fault

## State RAM Configuration for Discrete Modules

Depending on the selected module, the **State RAM** area of the **Memory** tab contains parameters for discrete input modules, discrete output modules or for discrete mixed modules. They are available if the option **Mixed topological and State RAM** is selected in the **Memory Management** area.

The figure below shows the **Memory** tab for a discrete mixed input / output module as an example:




The **State RAM** area contains the following parameters:

| Parameter | Description |
|---|---|
| **Format** | • **Bit**: Select this option for channel mapping in the discrete mode. This means that each channel is mapped to a `%I` (inputs) or `%M` (outputs).<br>• **Word**[1]: Select this option for channel mapping in the word mode. This means that a group of 8 channels is mapped to a `%IW` (inputs) or `%MW` (outputs). |
| **Coding** | This parameter is only available for channel mapping in word mode, that is if the parameter **Format** is set to **Word**.<br>• **Binary**: Select this option to use the binary integer format.<br>• **BCD**: Select this option to use the BCD coding format. |

| Parameter | Description |
|---|---|
| **Mapping** | This parameter allows you to define the **starting address**es for the inputs and outputs.<br>The following sizes of the memory areas are possible:<br>● For inputs in **Bit** format: `%I-1x`<br>● For inputs in **Word** format: `%IW-3x`<br>● For outputs in **Bit** format: `%M-0x`<br>● For outputs in **Word** format: `%MW-4x`<br>**NOTE:** The State RAM address starts at address 0, but the first available address for discrete and analog modules is `%I1`, `%M1`, `%IW1`, and `%MW1`. |

(1) When **Word** is selected as the **Format**, the LSB of the **Word** is mapped to:
- Input 0 on discrete input modules
- Output 0 on discrete output modules

**NOTE:** All channels of a discrete module must be configured. It is not possible to leave a group of channels un-configured.

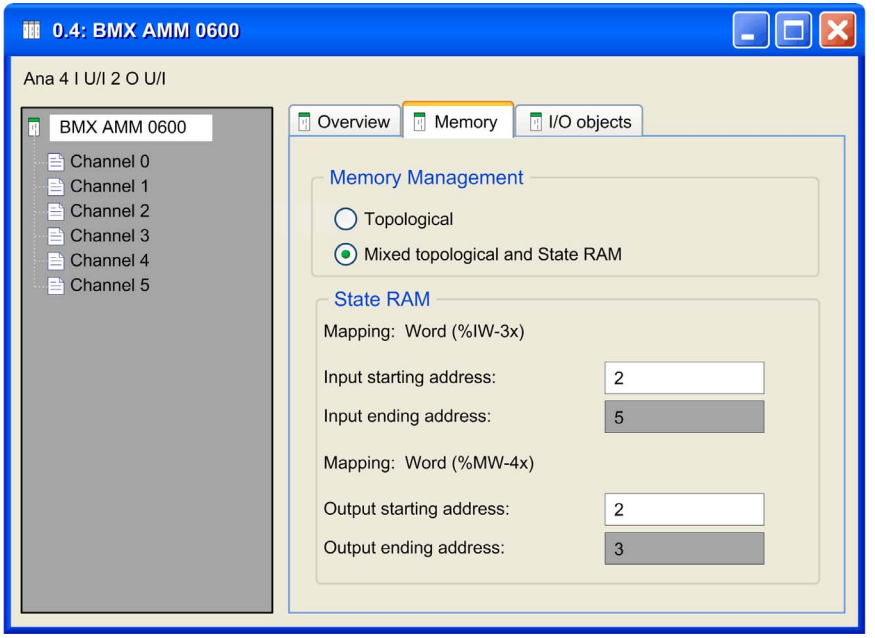
**NOTE:** You can use State RAM objects in event sections, but they are refreshed with the frequency of the task where they are declared.

**NOTE:** State RAM addresses are not checked during input. If any range overlapping is detected or any address exceeds the range of the State RAM, an appropriate message is displayed at the validation screen or during **Analyze**.

## State RAM Configuration for Analog Modules

Depending on the selected module, the **State RAM** area of the **Memory** tab contains parameters for analog input modules, analog output modules or for analog mixed modules. They are available if the option **Mixed topological and State RAM** is selected in the **Memory Management** area.

The figure below shows the **Memory** tab for a mixed analog input / output module as an example:




The **State RAM** area contains the following parameters:

| Parameter | Description |
|---|---|
| **Mapping** | This parameter allows you to define the **starting addresses** for the inputs and outputs.<br>The following sizes of the memory areas are possible:<br>● For inputs in **Word** format: `%IW-3x`<br>● For outputs in **Word** format: `%MW-4x` |

**NOTE:** State RAM addresses are not checked during input. If any range overlapping is detected or any address exceeds the range of the State RAM, an appropriate message is displayed at the validation screen or during **Analyze**.

# Chapter 31
# Type Library Browser

## Overview

This chapter describes the type library browser.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Access to the Type Library Browser | 1226 |
| Description of the Type Library Browser | 1228 |
| Configuration of Columns of the Type Library Browser | 1231 |
| Drag & Drop of FFBs in a program section | 1233 |

# Access to the Type Library Browser

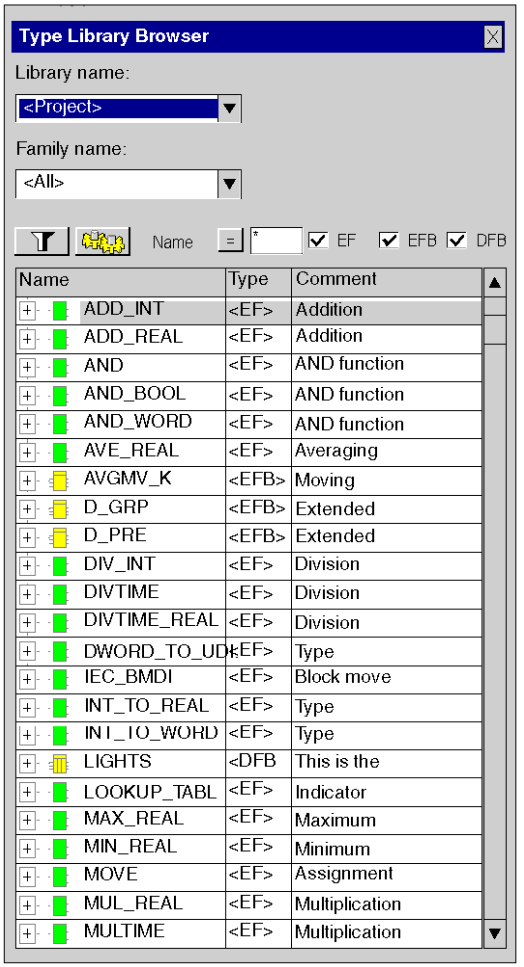
## At a Glance

Functions and function blocks (FFBs) can be dragged from the type library browser into a program section and stored (Drag & Drop).

## Accessing the Browser

| Step | Action |
|---|---|
| 1 | In the **Tools** menu, select the **Type Library Browser** option or click the **Type Library Browser** button  in the language editor toolbar.<br>(To close the **Type Library Browser** click the button again). |

| Step | Action |
|---|---|
| 2 | **Result:** The following window appears. |



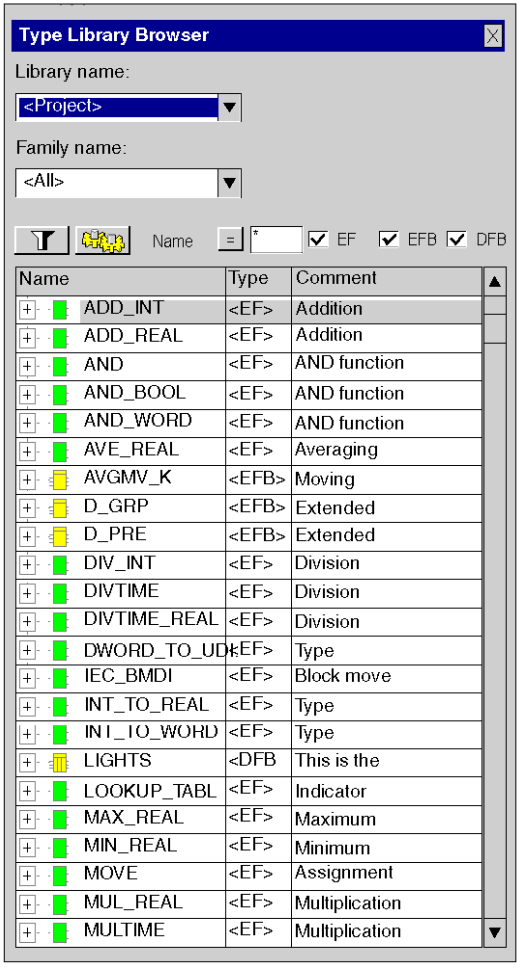
# Description of the Type Library Browser

## Calling the Type Library Browser

See *Access to the Type Library Browser, page 1226*.

## Structure of the window

Appearance:



Elements:

| Element | Description |
|---|---|
| **Library name** | The scope of the displayed function types and function block types is selected in this column:<br>● To display all function types and function block types used in the project, select **<Project>**.<br>● To display all function types and function block types of a library independently of the library assignment, select **<Libset>**.<br>● To display all function types and function block types of a library, select the desired library. |
| **Family name** | To display all function types and function block types of a library family, select the desired family. |
| Button  | Click this button to update the display corresponding to the filter term defined in the **Name** field. |
| Button  | Click this button to open a dialog box *(see page 412)* for defining the filters. |
| Button  | Click this button to invert the filter. The button changes from **=** to **<>** and vice versa. |
| **Name** (Text Box) | The name of the functions and function block types to search for can be entered in this text box. You can also use the joker ***** and **?**. |
| **EF** | When you enable this check box, Elementary Function types (EFs) are displayed. |
| **EFB** | When you enable this checkbox, Elementary Function Block types (EFBs) are displayed. |
| **DFB** | When you enable this checkbox, Derived Function Blocks (DFBs) are displayed. |
| **Name** (List) | This column displays the names of the function types and function block types and their formal parameters. |
| **Type** | This column displays the types of the function types and function block types. |
| **Comment** | This column displays the comments for the function types and function block types. |

Closing the **Type Library Browser** the currently selected **Library name**, **Family name** and type check boxes (**EF**, **EFB**, **DFB**) are saved. If you open the **Type Library Browser** again, these items will be preset.

## Shortcut menu structure

Additional different commands can be executed via the context menu of the Type Library Browser.

Shortcut menu commands:

| Menu command | Description |
|---|---|
| **Copy into Project** | This menu command is only available for functions and function block types that are not used yet in the project. Use this menu command to load the selected function and function block types into the project. Select via drop down list the **Library Name** which makes it quicker to access the type in the project. |
| **Expand All** | Use this menu command to display the formal parameters of the selected function and function block type. The same function is also available above the **+** symbol in front of the function block type names. |
| **Collapse** | Use this menu command to hide the formal parameters of the selected function and function component type. The same function is also available via the **-** symbol in front of the function block type names. |
| **Customize Columns...** | Using this menu command you can:<br>● define the columns *(see page 1231)* to be displayed and<br>● define the sequence *(see page 1231)* for the columns shown. |
| **Help on Type** | Use this button to open the help for the selected function or function block types shown. |
| **Data Properties** | Use this menu command to call the Data Properties *(see page 1197)* dialog box for the selected function and function block type. |

# Configuration of Columns of the Type Library Browser

## Introduction

You can use the **Column Configuration** dialog box to:

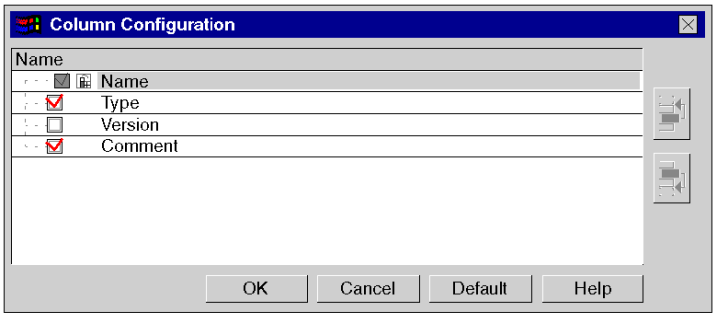
- define the columns that you want to display,
- define the sequence for the columns shown.

## Call the dialog box:

You can call up the **Column configuration** dialog box from the shortcut menu in the Type Library browser *(see page 1228)*.

## Call the dialog box

Representation:




Elements:

| Element | Description |
|---|---|
| **Name** | This checkbox is to show the names of the functions and function blocks in the Type Library Browser and cannot be unchecked. |
| **Type** | When you activate this checkbox, the Type Library Browser and the types (EFB, FB etc.) for the functions and function blocks are shown. |
| **Version** | When you enable this checkbox, the Type Library Browser and the versions of the functions and function blocks types are shown. |
| **Comment** | When you enable this checkbox, the Type Library Browser and the comments for the functions and function blocks types are shown. |

| Element | Description |
|---|---|
|  | Use this button to move the location of the selected attribute one place forward.<br>The location of the attribute **Name** cannot be modified. |
|  | Use this button to move the location of the selected attribute one place back.<br>The location of the attribute **Name** cannot be modified. |

# Drag & Drop of FFBs in a program section

## Procedure

To insert a function or function block (FFBs) using drag & drop, carry out the following steps:

| Step | Action |
|---|---|
| 1 | Open a program section |
| 2 | Open the Types Library Browser per drag & drop.<br>● the **Tools → Type Library Browser**menu command.<br>or<br>● press the key combination **Alt+3**. |
| 3 | Select the desired function/function block and hold down the left mouse button. |
| 4 | Drag the selected function/ function block types in the program section. |
| 5 | Release the mouse button.<br>**Result:**<br>● The FFB is inserted in the program section.<br>● In the **Tools → Options** dialog box on the **Data and Languages** tab, if the **Automatically assign a variable to a new graphical object** checkbox is selected, then the Function input assistant *(see page 1194)* is automatically opened for inputting the current parameters. |

# Chapter 32
# Search/Replace Tool

## Subject of this Chapter

This chapter presents the **Search / Replace** tool and how to use its main functions:

- the **Search** function is used to look for data,
- the **Replace** function is used to replace data.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Overview of the Search / Replace tool | 1236 |
| General Features for Search/Replace | 1238 |
| Searching for Data | 1239 |
| Replacing Data | 1240 |
| Search Toolbar | 1241 |
| Data Search Dialog | 1242 |
| Cross-References (Search Results) | 1244 |
| Cross-References (Replace) | 1247 |
| Tracing Variables / Addresses | 1249 |

# Overview of the Search / Replace tool

## Introduction

The **Search / Replace** tool provides you with the following functions:
- the **Search** function,
- the **Replace** function.

## Search Function

This function lets you **look for objects** such as:
- **instances** of elementary data types (EDT),
- **instances and types** of derived data (DDT / IODDT),
- **instances and types** of function block data (EFB / DFB),
- **instances and types** of data relating to Sequential Function Charts (SFC),
- **topological addresses** (`%M100`).

The search is carried out in the following areas of the project:
- sections that make up the application program,
- animation tables,
- operator screens,
- configuration.

## Replace Function

This function lets you **replace objects** such as:
- **instances** of elementary data types (EDT),
- **instances** of derived data types (DDT / IODDT),
- **instances** of function block data (EFB / DFB),
- **topological addresses** (`%M100`).

The replacement objects must be declared, except for direct addressing data (`%M`, `%MW`, etc.) and their type must correspond to that of the objects being replaced.

The replacement can done on the search results and initiated at the result window.

## Accessing the Data Search Dialog

To open the **Data Search** dialog

- click the **Data Search**  button in the toolbar or
- use the **Tools** → **Search / Replace** menu (**Alt+5**).

## Predefined Search

To start a predefined search

- from the **Data Editor** or
- from a language editor window,

select the object whose references you want to see and use **Initialize Search** from the context menu.

Another way is to select a variable in a language editor window. Doing so, the list box of the search toolbar *(see page 1241)* contains the variable name. Press **Enter** to start searching. You can also use 2 special buttons from the toolbar to search for reading/writing references *(see page 1241)*.

# General Features for Search/Replace

## Copy / Paste

You can use copy and paste in the following instances:

- copying a data item from any given editor in the programming software and pasting it into an entry field in the window,
- copying a data item from the window and pasting it into any given editor in the programming software,
- copying / pasting a data item within a window.

## Drag-and-Drop

You can use drag-and-drop for search.

Drag a data item from any given editor in the programming software and drop it at the end of the list of displayed references; the corresponding tree is created.

## Protection

The **Search** function is always available. Protection applies to:

- the references in the protected sections of the program, which are not displayed,
- the references in the protected sections of the DFBs, which are not displayed.

The **Replace** function complies with the protection applied to the different areas of the project:

- if the application field is in read-only, the **Replace** button is not available,
- if certain references are protected, they can not be selected for replacement.

## Online Mode

The use and behavior of **Search / Replace** in online mode are identical to the use and behavior in offline mode.

## Save Context

When you close the Search / Replace window, the following data is saved:

- the position and size of the window,
- the selection of search parameters,
- the 10 most recent entries in the drop-down menu,
- the current view (structural/functional and full/shortened/flat),
- the contents of the Unity Pro **Search / Replace** output window.

# Searching for Data

## Searching via Data Search Dialog

To search data using the **Data Search** dialog, you must perform the following actions:

| Step | Action |
|---|---|
| 1 | Open the **Data Search** dialog<br>● by clicking the **Data Search** button  in the toolbar or<br>● by using the **Tools** → **Search / Replace** menu (**Alt+5**) and pressing the **Data Search** button in the **Cross-References** window *(see page 1244)*.<br>**Result**:<br>The **Data Search Dialog** is displayed. |
| 2 | Specify the search options in the **Data Search Dialog** *(see page 1242)*. |
| 3 | Click the **Search** button.<br>**Result**:<br>The search results are displayed in the **Cross-References** window *(see page 1244)*. |

## Predefined Search

To start a predefined search

- from the **Data Editor** or
- from a language editor window,

select the object whose references you want to see and use **Initialize Search** from the context menu.

Another way is to select a variable in a language editor window. Doing so, the list box of the search toolbar *(see page 1241)* contains the variable name. Press **Enter** to start searching. You can also use 2 special buttons from the toolbar to search for reading/writing references *(see page 1241)*.

## Replacing Data

To access the data replace function, you must perform the following actions:

| Step | Action |
|---|---|
| 1 | After searching for data *(see page 1239)* the search results are displayed in the **Cross-References** dialog *(see page 1244)*. |
| 2 | In the **Cross-References** dialog click the  button to display the **Replace with** area of the dialog. |
| 3 | Enable the check boxes for those references of the search results which should be replaced. |
| 4 | ● Enter the replacement data (only data compatible with the data being searched for is tolerated)<br>● or click the **Down** arrow to select the replacement data out of a list providing the last 10 entries<br>● or click the **...** button to open a **Data Selection** dialog to select the instance or data to be replaced (only the instances compatible with the instance being searched for are displayed) |
| 5 | Click the **Replace** button to start the replacement operation in the preselected **Scope**. Please refer to **Data Search** dialog *(see page 1242)*. |

# Search Toolbar

## Introduction

The main toolbar of Unity Pro provides a **Data Search** button with a list box beside it to select or to enter a search string.




## Start Searching

To start searching the selected or entered search string press **Enter**.

The default search options are used.

The options can be changed by pressing the **Data Search** button left to the list box instead of pressing **Enter**.

The search results are displayed in the **Cross-References** dialog *(see page 1244)* and a jump to the first occurrence of the searched data is done.

## Data Search Dialog

To open the **Data Search** dialog *(see page 1242)* click the binoculars symbol.

## Searching for Reading/Writing References

To search for reading/writing references only, you can use

- the **Data Search** dialog *(see page 1242)* or
- two special buttons from the toolbar (see below).

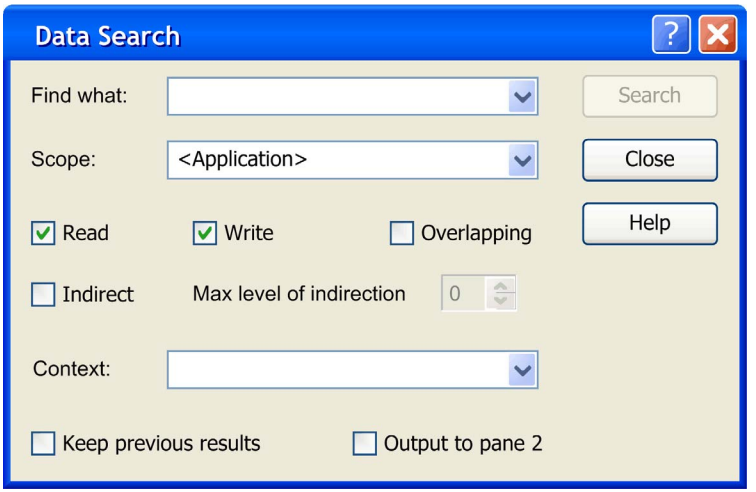
| Button | Description |
|---|---|
|  | Click this button to start searching for reading references only.<br>The current search options, defined in the **Data Search** dialog, are used but the **Write** option is ignored. |
|  | Click this button to start searching for writing references only.<br>The current search options, defined in the **Data Search** dialog, are used but the **Read** option is ignored. |

Using one of these 2 buttons initiates a direct jump to the first found occurrence of the reading/writing reference (for example, into the containing network).

If instead of a direct jump to the first found occurrence of the reading/writing reference the **Data Editor** is opened, either deactivate the Overlapping *(see page 1242)* option or click the **Go to next reference** button *(see page 1244)*.

# Data Search Dialog

## Representation




## Description

Elements of the **Search** tab

| Element | Description |
|---|---|
| **Find what** | Enter the complete variable name you want to search for.<br>Click the **Down** arrow to display a list providing the last 10 search strings.<br>Click the **...** button to open a **Data Selection** dialog to select the object you want to search for. |
| **Scope** | Select the scope where you want to search.<br>The different scopes are:<br>● application<br>● a specific DFB of the project<br>● all scopes (application + all DFBs) |
| **Search** button | Starts the search and opens search results window (**Cross-References**). |
| **Close** button | Closes the dialog. |
| **Help** button | Opens the Help for this dialog. |
| **Read** | Limits the search to reading references. |
| **Write** | Limits the search to writing references. |

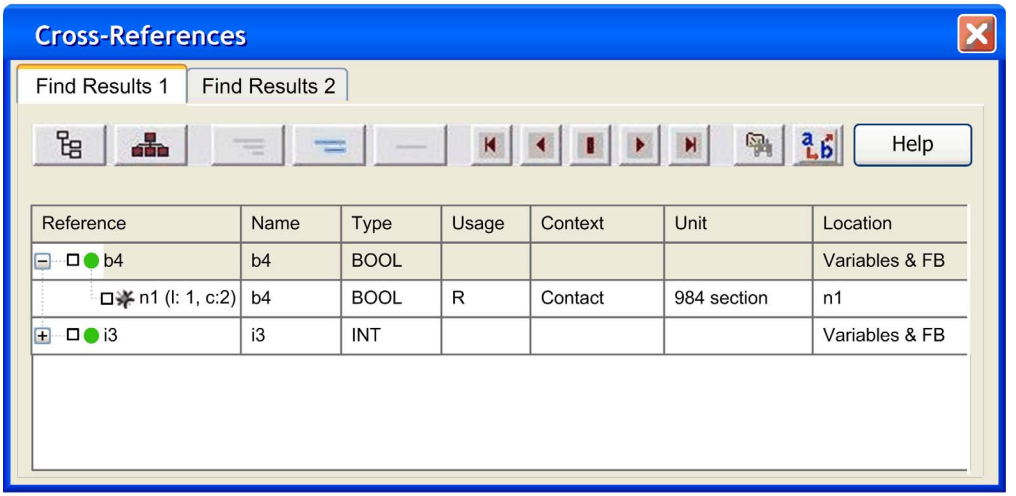
| Element | Description |
|---|---|
| **Overlapping** | Activate this check box to search for all complete and partial references. |
| **Indirect** | Activate this check box to search for sub-elements and direct parents of structured variables and arrays. |
| **Max level of indirection** | Defines the maximum level of indirect elements (sub-elements/parents) to be displayed. |
| **Context** | Limits the search to a specific context.<br>● **Contact**: variable attached to a contact<br>● **Coil**: variable attached to a LD/LL984 coil<br>● **SFC Object**: variable used with an SFC object<br>● **FFB / Subroutine**: variable passed to an FFB<br>● **ST / IL Statement**: variable used with an ST/IL statement<br>The context is displayed as an optional column in the search results. |
| **Keep previous results** | Activate this check box to keep the previous search results. Otherwise, the search results window (**Cross-References**) is cleared before the new search results are displayed. |
| **Output to pane 2** | Deactivate this check box to display the search results in the **Find Result 1** tab of the search results window (**Cross-References**).<br>Activate this check box to display the search results in the **Find Result 2** tab of the search results window (**Cross-References**). |

# Cross-References (Search Results)

## Introduction

The **Cross-References** search results dialog displays the results of a search.

## Representation




## Description

Elements of the **Cross-References** dialog

| Element | Description |
|---|---|
| **Find Result 1** tab<br>**Find Result 2** tab | Depending on the respective check box activated in the **Data Search** dialog *(see page 1242)*, the search results are displayed in one of the 2 tabs. |
|  | Displays the search results in a similar manner to the structural view in the **Project Browser**. |
|  | Displays the search results in a similar manner to the functional view in the **Project Browser**. |
|  | Displays the search results in **Full** form.<br>● Data declaration<br>● Path<br>● Data usage |
|  | Displays the search results in **Shortened** form.<br>● Data declaration<br>● Data usage |

| Element | Description |
|---|---|
|  | Displays the search results in **Flat** form.<br>● Data usage |
|  | Selects the previous reference of the search results. |
|  | Selects the next reference of the search results. |
|  | Jumps directly to the currently selected found occurrence of the reference. |
|  | Jumps directly to the previous found occurrence of the reference (for example into the containing network). |
|  | Jumps directly to the next found occurrence of the reference. |
|  | Opens the **Data Search** dialog. |
|  | Displays/hides the **Replace with** area of the dialog *(see page 1247)*. |
| **Reference** | This column displays the name of the instance or type (DDT/IODDT/EFB/DFB) of the data searched for. The details displayed depend on the display settings described above (**Full**, **Shortened**, **Flat**). |
| **Type** | This column displays<br>● type and reference of the data searched for,<br>● programming language used in the sections, subroutines, events, etc. that contain references to the data searched for. |
| **Usage** | This column displays<br>● the nature of the references to the data searched for (read only, write only, read / write),<br>● the data search area. |
| **Context** | Specific context as defined in the Data Search Dialog *(see page 1242)*. |
| **Unit** | This column displays the unit of the **Location**. |
| **Location** | This column displays the location of the data searched for in a short form. |

## Sorting Order

When the search results are displayed in **Full** or **Shortened** form sorting can only be done by **Reference** or **Type**.

When the search results are displayed in **Flat** form sorting can also be done by reference related columns (**Usage**, **Context**, **Unit** and **Location**).

The search results are first sorted by their path (Program, Task, Segment, Section/Network).

The search results of sections/networks are sorted by a second key, depending on the kind of the section/network.

- References within IEC language sections (FBD, LD, SFC, ST, IL) are sorted by row and column.
- References within LL984 networks are sorted by column and row.

You can re-sort the search results by clicking on a specific column.

Re-sorting by multiple keys is not supported.

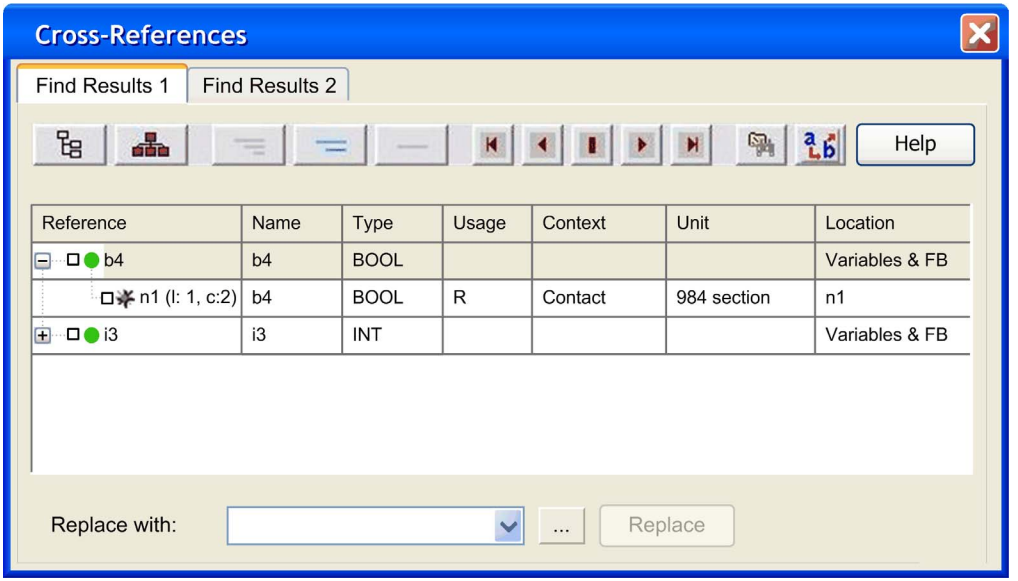
# Cross-References (Replace)

## Introduction

The **Cross-References** (Search Results) dialog *(see page 1244)* displays the results of a search.

To display the **Replace with** area of the dialog use the  button.

## Representation



## Description

Elements of the **Replace with** area of the dialog

| Element | Description |
|---|---|
| **Replace with** | To display this area of the dialog use the  button.<br>● Enable the check boxes for those references of the search results which should be replaced.<br>● Enter the replacement data (only data compatible with the data being searched for is tolerated),<br>● or click the **Down** arrow to display a list providing the last 10 entries,<br>● or click the **...** button to open a **Data Selection** dialog to select the instance or data to be replaced (only the instances compatible with the instance being searched for are displayed). |
| **Replace** button | Starts the replacement operation in the preselected **Scope**. Please refer to **Data Search** dialog *(see page 1242)*. |

# Tracing Variables / Addresses

## Introduction

Using the tracing feature, you can locate the occurrences of a variable/address in an application.

## Tracing

| Step | Action |
|---|---|
| 1 | Select an object in a language editor window.<br>**Result**:<br>The address attached to the object is displayed in the edit field of the Search Toolbar *(see page 1241)*. |
| 2 | Start a search either<br>● by pressing the **Enter** key or<br>● by means of one of the 2 special buttons next to the edit field of the Search Toolbar (Searching for Reading/Writing References *(see page 1241)*).<br>**Result**:<br>● The search results are displayed in the **Cross-References** dialog *(see page 1244)*.<br>● A direct jump is initiated to the first found occurrence of the reading/writing reference (for example into the containing network).<br>● A new automatic **Search_"<Search String>"** bookmark is created (to be able to jump back later on). This bookmark is inserted into the bookmarks list like any other manually created bookmark. Automatically created bookmarks *(see page 580)* are displayed in gray and can not be renamed. |
| 3 | Double-click a search result, displayed in the **Cross-References** dialog, to jump to the respective read/write location. |
| 4 | With the bookmarks feature you can mark one or more of the shown locations in the language editor. These bookmarks can be used later on to jump to from the bookmarks window. |
| 5 | From the bookmarks window you can also select the automatic **Start of Search** bookmark for going back to the original starting position of the search.<br>**NOTE:** You can delete automatically generated bookmarks via the bookmarks window. |

# Chapter 33
# User's Function Blocks

## Overview

This chapter describes how to use the user's function blocks: DFB.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 33.1 | DFB Type | 1252 |
| 33.2 | DFB Instance | 1272 |
| 33.3 | How to Create Nested DFBs | 1284 |

# Section 33.1
# DFB Type

## Purpose of this section

This section describes the types of DFB.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| DFB Type | 1253 |
| Creating a DFB Type | 1255 |
| Configuration of a DFB Type | 1256 |
| Programming a DFB type | 1259 |
| Deleting a DFB type element | 1262 |
| Archiving a DFB Type | 1263 |
| How to protect a DFB type | 1264 |
| Deleting a DFB Type | 1266 |
| How to access the attributes of a DFB type | 1267 |
| Authorized DFB Type Modifications | 1269 |

# DFB Type

## General information about DFBs

A **DFB** (Derived Function Block) is a user function block that has been customized to take the specific nature of your project into consideration. It can be stored in the user-defined library.

To use a DFB in your application, you must:

- Create a user's function block model called DFB type *(see Unity Pro, Program Languages and Structure, Reference Manual )*,
- Use the available copies of the model called **DFB instances** *(see Unity Pro, Program Languages and Structure, Reference Manual )* in your project.

## Online Help for DFB Types (Help on Type)

It is possible to link an HTML help file to each DFB in the user-defined library.

- This file must have a name that is identical to the linked DFB and the extension must be **.htm*.
-
  - For Windows XP this file must be located in the directory:
    *C:\Documents and Settings\All Users\Application Data\Schneider Electric\Unity Pro\CustomLibset/Vx.x/**Language***.
  - for Windows Vista this file must be located in the directory:
    *C:\ProgramData\Schneider Electric\Unity Pro\CustomLibset/Vx.x/**Language***.

***Language*** is named ***ENG***, ***FRE***, ***GER***, ***ITA***, ***SPA*** or ***CHI*** according to the language desired.

Creating **Help on Type**

| Step | Action |
|---|---|
| 1 | Create an HTML help file for your DFB (for example with any kind of HTML editor).<br>**Note:** The HTML file name must be exactly the same as the name of your DFB. |
| 2 | Copy this file into the respective language folder (***ENG***, ***FRE*** ...). |
| 3 | Create an additional folder called ***HELP*** in the same directory as the language folders. |
| 4 | Copy all the files (for example graphics) you are referencing in your HTML file to the ***HELP*** folder. |
| 5 | Install your user-defined library in Unity Pro.<br>**Result:**<br>All the files will be copied to the ***Libset*** directory and the HTML help file will be launched when you click the **Help on Type** button. |

## Accessing a DFB Type

To create a DFB type, you must use the program **Unity Pro**.

DFB types are accessed via the **Derived FB Types** directory in the project browser.

**NOTE:** You can also access the DFB types definition tool via the directory **Variables & FB instances**.

## Accessing DFB Types Via the Derived FB Types Directory

To access the DFB types via the **Derived FB Types** directory, do the following:

| Step | Action |
|---|---|
| 1 | In the structural view of the project browser, right-click on the directory **Derived FB Types**.<br>**Result**: A shortcut menu is opened. |
| 2 | Activate the **Open** command.<br>**Result**: The data editor window is opened. It is positioned directly under the **DFB Types** register tab and shows the list of existing DFBs. A double-click on the first available line indicated by an arrow provides access to the creation of a new DFB type. |

# Creating a DFB Type

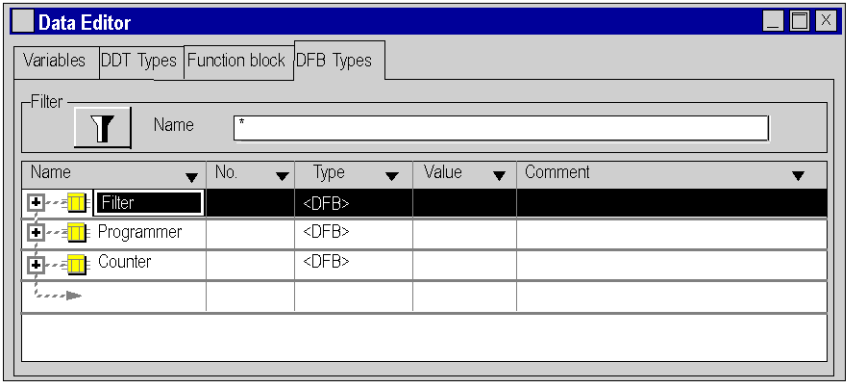
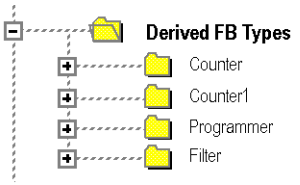
## How to create a DFB type

To create a DFB type *(see Unity Pro, Program Languages and Structure, Reference Manual )*, you must do the following:

- Enter a name on the empty line to create the empty structure of the DFB type,
- configure the DFB type,
- program the DFB Type (in the codes section).

## Creating a DFB type

Do the following to create the empty structure of a DFB Type:

| Step | Action |
|---|---|
| 1 | Open the **DFB types** *(see page 1253)* register tab in the data editor.<br>**Result**: The list of existing DFBs is displayed.<br>  |
| 2 | Select the first empty **Name** cell (shown by an arrow) and enter the parameter or variable name and confirm with **Enter**.<br>**Result**: The empty structure of the DFB type is now created. This new DFB added to the list of existing DFBs. It also appears in the directory tree **Derived FB Types**.<br>  |

# Configuration of a DFB Type

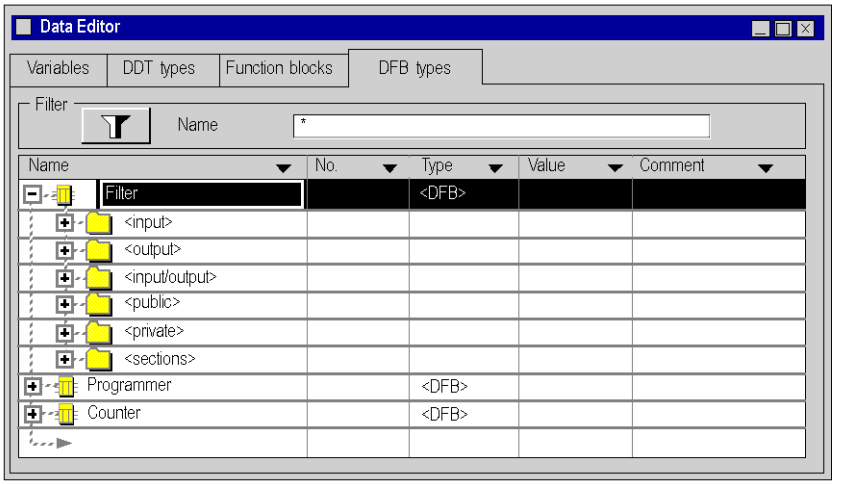
## How to Configure a DFB Type

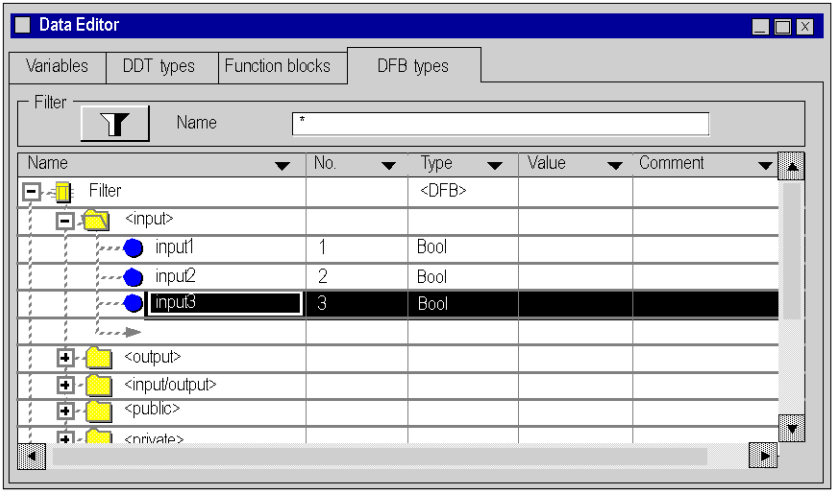
To configure a DFB type *(see Unity Pro, Program Languages and Structure, Reference Manual )* you must:

- input the parameters of the DFB: input, output and input/output
- input the DFB variables: public or private
- Input the DFB description.

## Input of the Parameters and Variables

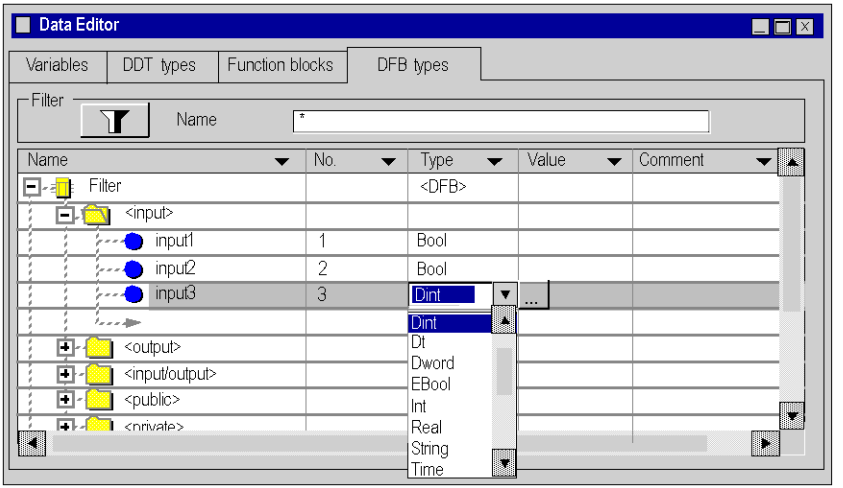
Carry out the following actions to input the parameters and the variables for a DFB:

| Step | Actions |
|---|---|
| 1 | Go to the **DFB types** *(see page 1253)* tab of the data editor.<br>**Result**: the list of existing DFBs is displayed on the screen. |
| 2 | Click on the **+** sign positioned in front of the name of the DFB type to be configured.<br>**Result**: the DFB type is expanded, which gives access to its various directories (**input**, **output**, etc.).<br>  |
| 3 | Click on the **+** sign of the directory that you want to expand: **input**, **output**, **input/output**, **public**, **private**. |

| Step | Actions |
|---|---|
| 4 | Select the first empty **Name** cell (indicated by an arrow) and enter the name of the parameter or the variable, then confirm using **Enter**.<br>**Result**: the data is created with a `BOOL` type by default (or with the last type selected).<br>  |
| 5 | Repeat steps 4 and 5, for all data to be input. |

## Parameter Type Modification

Carry out the following actions to modify the type of DFB parameter:

| Step | Actions |
|---|---|
| 1 | Double click on the **Type** cell associated with the desired parameter.<br>**Result**: you have access to the list of available types *(see page 1189)*.<br>  |
| 2 | Select a new type and confirm using **Enter**.<br>**Result**: data is modified with the new type.<br>**Note:**The MFB can be programmed in a DFB. To do this setup the type of an input by typing AXIS_REF. (this type name is not proposed in the list box but is a valid type for a DFB input).<br>**Note:** The IODDT related to CANopen devices for Modicon M340 cannot be used as a DFB I/O parameter. During the analyse/build step of a project, the following message:"This IODDT cannot be used as a DFB parameter" advises the limitations to the user. |

## New Pin Number Allocation

Carry out the following actions to allocate a new pin number to a DFB parameter (when parameter is created, a default pin number is allocated):

| Step | Actions |
|---|---|
| 1 | **Only** for **input**, **output** and **input/output** directories:<br>if you want to allocate a new pin number to every element of data, double click on the relevant **No.** cell.<br>**Result**: the arrow is replaced by a data entry field. |
| 2 | Enter the desired new pin number then **Enter** to validate.<br>**Result**: the data is associated with the selected pin number. |

# Programming a DFB type

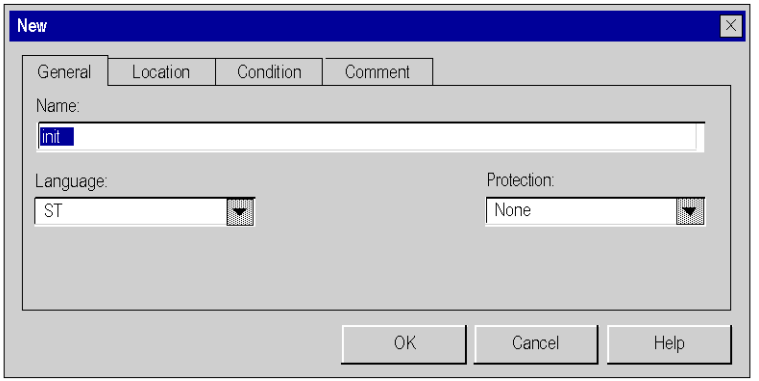
## How to program a DFB type

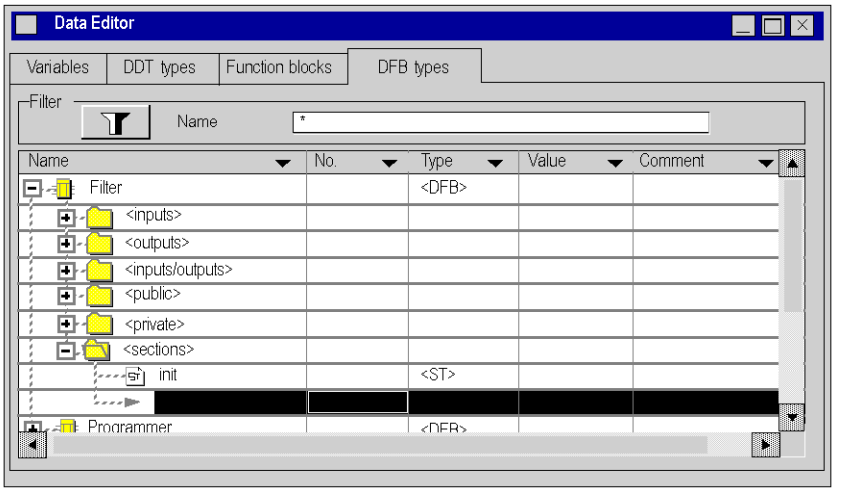
To program a DFB type *(see Unity Pro, Program Languages and Structure, Reference Manual )* you must:

- create the DFB section(s),
- enter the program code for each section.

## Creating a section

Carry out the following actions in order to create a DFB section:

| Step | Action |
|---|---|
| 1 | Go to the **DFB types** *(see page 1253)* tab of the data editor.<br>**Result**: the list of existing DFBs is displayed on the screen. |
| 2 | Expand the DFB type that you would like to program. |
| 3 | Expand the **sections** directory by clicking on **+**. |
| 4 | Select the **Name** cell and enter the name of the section.<br>Confirm with **Enter**.<br>**Result**: a dialog box is displayed on the screen to allow you to create a new section.<br>  |
| 5 | Select the programming language desired in the drop-down menu in the **Language** field. |

| Step | Action |
|---|---|
| 6 | Click **OK** to confirm the command.<br>**Result**: the section appears in the **sections** directory.<br>  |
| 7 | If the **sections** directory already contains sections and you want to create a new one, expand the directory and select the first empty **Name** cell.<br>**Result**: the dialog box used to enter section parameters is displayed on the screen. |
| 8 | Enter the section parameters and confirm with **OK**.<br>**Result**: the new section appears in the **sections** directory. |

## Entering the program

Carry out the following actions in order to enter the program for the section:

| Step | Action |
|---|---|
| 1 | Expand the **sections** directory of the DFB whose code you would like to enter.<br>**Result**: the list of sections is displayed on the screen. |
| 2 | Right-click on the name of the section to program.<br>**Result**: the contextual menu appears. |
| 3 | Activate the **Open** command.<br>**Result**: the section programming screen is displayed on the screen, in the selected language. |
| 4 | Enter the section program *(see page 451)*. |

## Programming restrictions

All language instructions are allowed, except instructions that use input/output module variables (READ_STS, READ_PARAM, WRITE_CMD, etc..).

Connection to a label (JUMP) is only possible within the same section.

You may not use the following objects:

- input/output objects (%I, %Q, etc.),
- the application's global objects (%MW, %KW, ...), except system bits and words %S and %SW.

**NOTE:** When the **Create variable** option in the contextual menu is used from a DFB section, the variable created is a private variable.

**NOTE:** For PL7 Function blocks (Obsolete Library): it is recommended not to use the PL7_••• function blocks inside DFBs because you are not allowed to create new instances in online mode.

# Deleting a DFB type element

## Deleting an element

To delete an element, do the following:

| Step | Action | | |
|---|---|---|---|
| 1 | Open the **DFB types** *(see page 1253)* tab of the data editor and expand the DFB type from which you want to remove an element. | | |
| 2 | Expand the directory that contains the element you want to delete. | | |
| 3 | **Either:**<br>Select the element that you want to delete. | | **Or:**<br>Right-click on the element that you wish to delete.<br>**Result**: A contextual menu appears. |
| 4 | Press **Delete**.<br>**Result**: The item is removed from the directory. | | Activate the command **Delete**.<br>**Result**: The item is removed from the directory. |

# Archiving a DFB Type

## Introduction

After having created a DFB type *(see Unity Pro, Program Languages and Structure, Reference Manual )*, you can back it up in a user library *(see page 355)* if the following conditions are met:

- You are using Unity Pro software,
- The computer is in offline mode,
- You have the required access rights,
- The DFB Type to back up is compatible (the nested DFBs and DDTs are also compatible),
- The DFB type name doesn't already exist in the library (the name of the nested DFBs and DDTs must also not already exist in the library).

When a DFB Type is backed up in a library, it can be used if the following conditions are met:

- You are using Unity Pro software,
- The computer is in offline mode,
- You have the required access rights.

## Archiving a DFB Type

To archive a DFB type in a user library, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **DFB types** *(see page 1253)* register tab in the data editor. |
| 2 | Right-click on the DFB type that you wish to back up.<br>**Result**: A shortcut menu is opened. |
| 3 | Activate the **Copy into library** command. |
| 4 | Select the library and the destination family.<br>Click **OK** to confirm. |

# How to protect a DFB type

## At a Glance

There are 4 levels of protection for a DFB Type:

- **Read-only**: The DFB type parameters directories (**inputs**, **outputs**, **inputs/outputs**, **public**, **private** and **sections**) are in read only format,
- **Version protection**: The DFB type is not protected except the DFB version number.
- **No Read & Write**: The DFB types parameters directories **private** and **sections** are not displayed. All the other the DFB type parameters directories (**inputs**, **outputs**, **inputs/outputs** and **public**) can be accessed from the data editor in read only format.
- **No protection**: The DFB type is not protected.

**NOTE:** If a DFB uses a DDT, you can modify the DDT type even if the DFB is protected.

## Protection of a DFB type

To protect a DFB Type, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **DFB types** *(see page 1253)* register tab in the data editor. |
| 2 | Select the DFB Type to protect. |
| 3 | Select the menu command **Edit** → **Properties**.<br>**Result**: The respective screen is displayed. |
| 4 | Select the protection level from the drop-down menu. |
| 5 | After selecting, enter your password in the **Entry** field (maximum 8 characters). |
| 6 | Confirm your password in the **Confirmation** field. |
| 7 | Check the **Password Encrypted** check box if an enhanced password protection is required.<br>**NOTE:** A DFB Type with a crypted password can not be edited with a Unity Pro version lower than 4.1. |
| 8 | Click **OK** to confirm. |

### Modification of the protection level

To change a DFB Type protection level, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **DFB types** *(see page 1253)* register tab in the variables editor. |
| 2 | Select the DFB Type to change the level of protection for. |
| 3 | Select the menu command **Edit** → **Properties**.<br>**Result**: The respective screen is displayed. |
| 4 | Select the attribute **Protection** and click on the related box in the **Value** column.<br>**Result**: A drop-down menu appears. |
| 5 | Select the new protection level from the drop-down menu **None**, **Version Protection**, **Read Only**, **No Read&Write**). |
| 6 | After selecting, enter the current password if the DFB was already protected. |
| 7 | Click **OK** to confirm. |

### Modify Password

To change a DFB type password, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **DFB types** *(see page 1253)* register tab in the variables editor. |
| 2 | Select the DFB type to change the password for. |
| 3 | Select the menu command **Edit** → **Properties**.<br>**Result**: The respective screen is displayed. |
| 4 | Enable the attribute **Protection** by clicking on the **+** sign. |
| 5 | Select the attribute **Password** and click on the related box in the column **Value**.<br>**Result**: The **Change password** dialog box is opened. |
| 6 | Enter the current password in the **Enter password** field. |
| 7 | Enter your new password in the **Entry** field (maximum 8 characters). |
| 8 | Confirm your new password in the **Confirmation** field. |
| 9 | Check the **Password Encrypted** check box if an enhanced password protection is required.<br>**NOTE:** A DFB Type with a crypted password can not be edited with a Unity Pro version lower than 4.1. |
| 10 | Click **OK** to confirm. |

# Deleting a DFB Type

## Deleting a DFB type

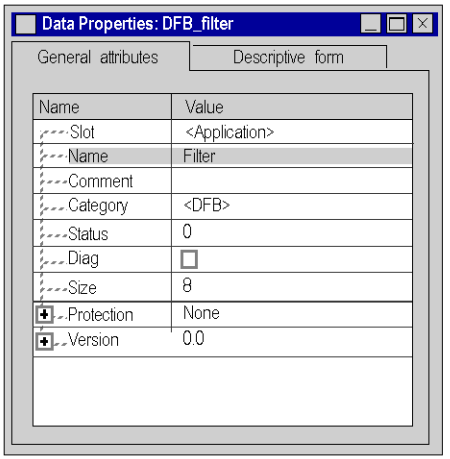
To delete a DFB type, do the following:

| Step | Action | | |
|---|---|---|---|
| 1 | Open the **DFB types** *(see page 1253)* tab in the data editor.<br>**Result**: The list of existing DFBs is displayed. | | |
| 2 | **Either:**<br>Select the DFB that you want to delete. | | **Or:**<br>Right-click on the DFB that you wish to delete.<br>**Result**: A contextual menu appears. |
| 3 | Press **Delete**.<br>**Result**: The item is removed from the list. | | Activate the command **Delete**.<br>**Result**: The item is removed from the list. |

# How to access the attributes of a DFB type

## Accessing the Attributes of a DFB Type

You should perform the following steps in order to access the attributes of a DFB type:

| Step | Action |
|---|---|
| 1 | Go to the **DFB Types** *(see page 1253)* tab of the data editor. |
| 2 | Right-click the DFB type whose attributes you would like to display.<br>**Result**: the contextual menu appears. |
| 3 | Enable the **Data Properties** command.<br>**Result**: The corresponding dialog box appears on the screen.<br>  |
| 4 | If you would like to associate an explanatory text with the definition of the DFB type (several lines of text with hyperlinks), select the **Descriptive form** tab.<br>**Result**: the **Descriptive form** dialog box appears *(see page 1849)*. |

## Attribute List

The attributes of a DFB type are the following:

- **Slot**,
- **Name**,
- **Comment**,
- **Category**,
- **State** (**1 if the modification is in progress** or **O if the analysis has been completed**),
- **Diag**.
- **Size**,
- **Protection**,
  - **Password**,
- **Version**,
  The version number is automatically incremented after a modification of the DFB type.
  - **Date of the last analysis**,

**NOTE:** All of the attributes of a DFB type can be accessed in **read** mode.
Only the following attributes can also be accessed in **write** mode:

- **Name**,
- **Comment**,
- **Protection**,
  - **Password**,
- **Diag**.

# Authorized DFB Type Modifications

## At a Glance

The modifications possible on a DFB type *(see Unity Pro, Program Languages and Structure, Reference Manual )* are defined by the access rights authorized by your user profile.

The following table summarizes the authorized modifications when you have a default user profile:

| Modification | Read only | Operation | Adjustment | Debugging | Programming |
|---|---|---|---|---|---|
| **DFB type in offline mode** | No | No | No | No | Yes |
| **DFB type in online mode** | No | No | No | No | Yes |
| **DFB instance in offline mode** | No | Yes | Yes | Yes | Yes |

**NOTE: In offline mode** any kind of modification of a DFB Type is authorized on condition that a non-blocking protection level is used.

## Authorized DFB type modifications

The following table shows the operations that you can perform with DFB types in online mode:

| Action | Non-used DFB type | Used DFB type |
|---|---|---|
| Create a DFB type | Yes | No |
| Delete a DFB type | Yes | No |
| Modify the name | Yes | No |
| Change Protection Level | Yes | No |
| Modify the comment | Yes | Yes |
| Modify the description | Yes | Yes |
| Export a DFB type | No | No |
| Import a DFB type | No | No |
| Make a back up to a library of a DFB Type | No | No |
| Use a DFB type from a library | Yes | Yes |

### Authorized modifications on the input/output parameters

The following table shows the operations that you can perform for the input/output parameters of a DFB Type in online mode:

| Action | Non-used DFB type | Used DFB type |
|---|---|---|
| Create a parameter | Yes | No |
| Delete a parameter | Yes | No |
| Move a parameter | Yes | No |
| Modify the name | Yes | No |
| Change type | Yes | No |
| Modify the comment | Yes | Yes |
| Change the initial value | Yes | Yes |
| Modify the other attributes | Yes | No |

### Authorized modifications on public variables

The following table shows the operations that you can perform on public variables of a DFB type in online mode:

| Action | Non-used DFB type | Used DFB type |
|---|---|---|
| Create a variable | Yes | Yes |
| Delete a variable | Yes | No |
| Modify the name | Yes | No |
| Change type | Yes | No |
| Modify the comment | Yes | Yes |
| Change the initial value | Yes | Yes |
| Modify the other attributes | Yes | No |

### Authorized modifications on private variables

The following table shows the operations that you can perform on private variables of a DFB type in online mode:

| Action | Non-used DFB type | Used DFB type |
|---|---|---|
| Create a variable | Yes | Yes |
| Delete a variable | Yes | No |
| Modify the name | Yes | No |
| Change type | Yes | No |
| Modify the comment | Yes | Yes |
| Change the initial value | Yes | Yes |
| Modify the other attributes | Yes | No |

### Authorized modifications of the sections

The following table shows the operations that you can perform on the sections of a DFB Type in online mode:

| Action | Non-used DFB type | Used DFB type |
|---|---|---|
| Create a section | Yes | Yes |
| Delete a section | Yes | Yes |
| Move a section | Yes | Yes |
| Modify the name | Yes | No |
| Modify the comment | Yes | Yes |
| Modify the code | Yes | Yes |

# Section 33.2
# DFB Instance

## Purpose of this section

This section describes DFB instances. DFB instances management is identical to the management of EFB instances *(see page 359)*.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| DFB Instance | 1273 |
| Creating a DFB Instance | 1274 |
| Creating several DFB Instances | 1276 |
| Deleting a DFB instance | 1277 |
| Operations on DFB Instances | 1278 |
| Modifying the Attributes of a DFB Instance | 1280 |
| Authorized DFB Instance Modifications | 1283 |

# DFB Instance

## At a Glance

A DFB instances *(see Unity Pro, Program Languages and Structure, Reference Manual )* is a copy of a validated DFB type *(see Unity Pro, Program Languages and Structure, Reference Manual )* that can be used for programming.

You can create **multiple instances** of a same DFB type. In this case the inputs/outputs parameters and the variables (public and private) are duplicated. The DFB type code is not duplicated.

You can use the **same instance** of the DFB multiple times in your application. This operation is nevertheless **strongly discouraged**.

The DFB instance name must be different from the name of a Unity Pro reserved word, a symbol, or from an elementary function (EF) name, or DFB type name.

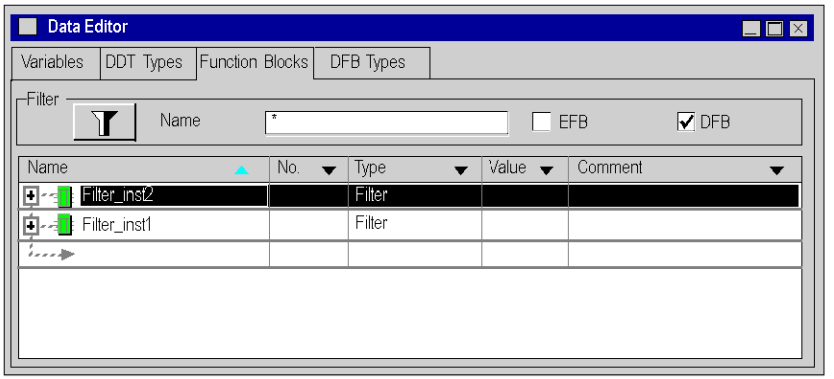
## How to access to a DFB instance

To access a DFB instance, you must use the program Unity Pro.

DFB instances are accessed via the **Variables & FB instances** directory in the project browser.

## Accessing DFB instances via the Derived FB instances directory

To access the DFB instances via the **Derived FB instances** directory, you must do the following:

| Step | Action |
|---|---|
| 1 | From the project browser structural view, expand the directory**Variables & FB instances**. |
| 2 | Double-click on the folder **Derived FB instances**.<br>**Result**: The data editor window is opened. It is automatically placed under the **Function Blocks** register tab. The list of instances already in the system is displayed.<br>  |

# Creating a DFB Instance

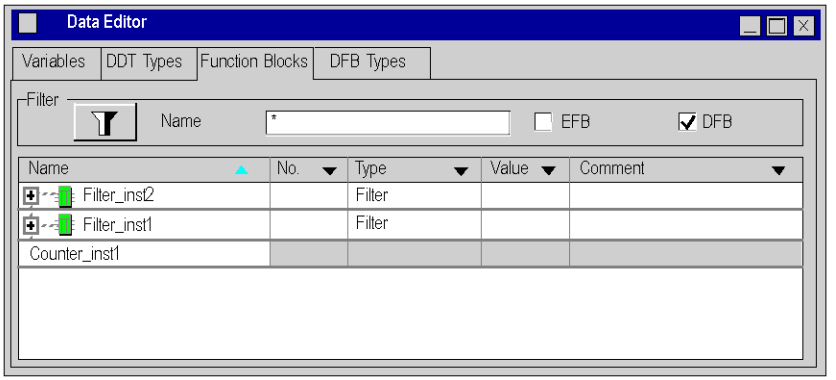
## At a Glance

There are two possible ways in which you can create a DFB instance *(see Unity Pro, Program Languages and Structure, Reference Manual )*:

- by inserting a new instance in the function block list,
- by inserting an instance in an editor.

## Inserting a new DFB instance in the data editor

To create a new DFB instance, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* register tab in the data editor. |
| 2 | Select the first empty **Name** cell (shown by an arrow) and enter name of the new DFB instance and confirm with **Enter**.<br>  |
| 3 | Then change the associated attributes *(see page 1280)*. |

## Inserting a new DFB instance in a language editor

To insert a new DFB instance in a language editor, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the language editor (the LD editor for example). |
| 2 | Click on an empty zone in the language editor.<br>**Result**: A shortcut menu is opened. |
| 3 | Activate the **Data Selection** command. |
| 4 | In the drop-down list, select the DFB type of which you wish to insert an instance. |
| 5 | Confirm with **Enter**.<br>**Result**: The mouse cursor is shown as a function block. |
| 6 | Click on the location where you wish to insert the DFB instance.<br>**Result**: A new DFB instance is created with a default name. It is inserted in the language editor and created in the data editor at the same time. |

# Creating several DFB Instances

## At a Glance

Several DFB instances can be created at once using the import function.

## Creation of several DFB instances by importing a source file

Several DFB instances *(see Unity Pro, Program Languages and Structure, Reference Manual )* can be created by importing *(see page 1719)* a variables exchange file (.xsy or .txt format). If you have chosen to export the variables with the associated DFB types, the latter are imported.

# Deleting a DFB instance

## Deleting a DFB instance

To delete a DFB instance, you must do the following:

| Step | Action | | |
|---|---|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* tab in the data editor. | | |
| 2 | **Either:**<br>Select the DFB instance that you want to delete. | | **Or:**<br>Right-click on the DFB instance that you wish to delete.<br>**Result**: A contextual menu appears. |
| 3 | Press **Delete**.<br>**Result**: The item is removed from the list. | | Activate the command **Delete**.<br>**Result**: The item is removed from the list. |

# Operations on DFB Instances

## At a Glance

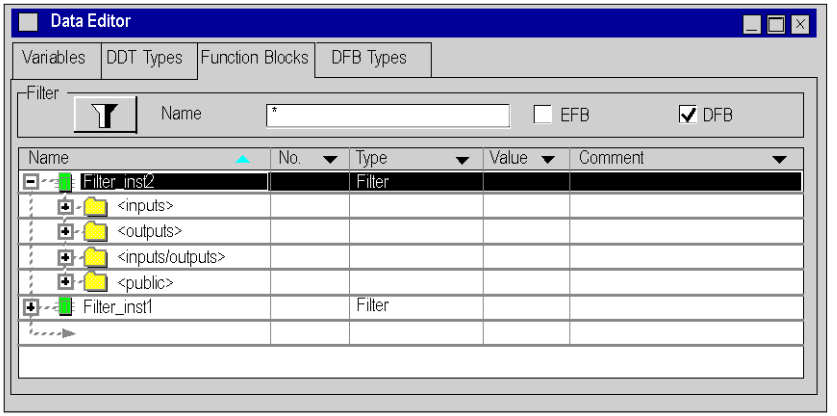
For each DFB instance *(see Unity Pro, Program Languages and Structure, Reference Manual )*, the **Function Blocks** register tab in the data editor provides one line of information by default: Instance name, DFB type, comment, etc. From this screen you can also:

- Apply a filter on the list of instances *(see page 412)*,
- View the elements of a DFB instance,
- View the attributes of a DFB instance,
- Sort the DFB instances according to their attributes.

**NOTE:** The DFB instance structure cannot be modified using the **Function Blocks** register tab (name and type of the elements). However, the attributes of the instance itself can be modified (name, type and comments of the DFB instance).

## View the elements of a DFB instance

To view the elements of a DFB instance, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* register tab in the data editor. |
| 2 | Open the DFB instance of which you want to view the elements.  |

## View the elements of a DFB instance

To view the attributes of a DFB instance, you must do the following:

| Step | Action | | |
|---|---|---|---|
| 1 | **Either:**<br>Open the **Function Blocks** *(see page 1273)* register tab in the data editor. | | **Either:**<br>Open the DFB instance to gain access to its elements. |
| 2 | Right-click on the DFB instance whose attributes you wish to view.<br>**Result**: A shortcut menu is opened. | | Right-click on the element whose attributes you wish to view.<br>**Result**: A shortcut menu is opened. |
| 3 | Activate the **Properties** command.<br>**Result**: The related dialog box is opened. | | |

## Attributes list

The specific DFB instance attributes are:

- **Instance name**,
- **Comments**,
- **Type**,
- **Category**,
- **Size**,
- **Diag**,
- **Used (frequency of use in program)**,
- **Customize**.

**NOTE:** All the DFB instance attributes are available in **read/write** mode except the **Category** and the **Used** attributes of instances, which are available **in read only**.

## Sorting DFB instances

To change the classification criteria, do the following:

| Step | Action |
|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* register tab in the data editor. |
| 2 | Click on the title of the column that you want for the first criteria.<br>**Result**: The DFB instance classification criteria become:<br>● Firstly, a classification by alphabetical order in the column on which you have clicked,<br>● then, a classification by alphabetical order by name.<br>**Note**: The DFB instances are classified by name alphabetically as default. |

# Modifying the Attributes of a DFB Instance

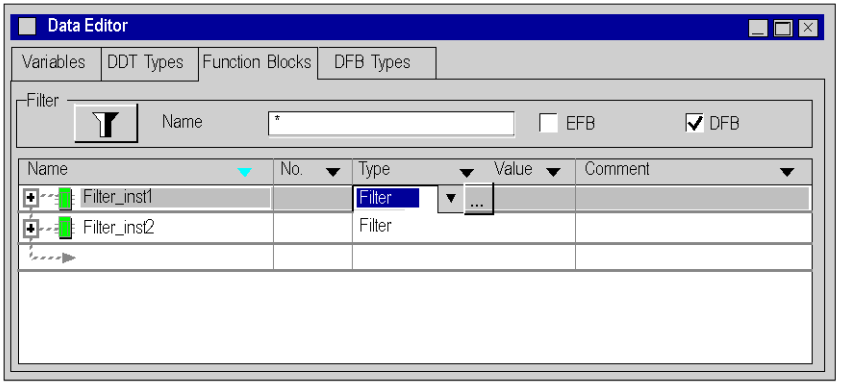
## At a Glance

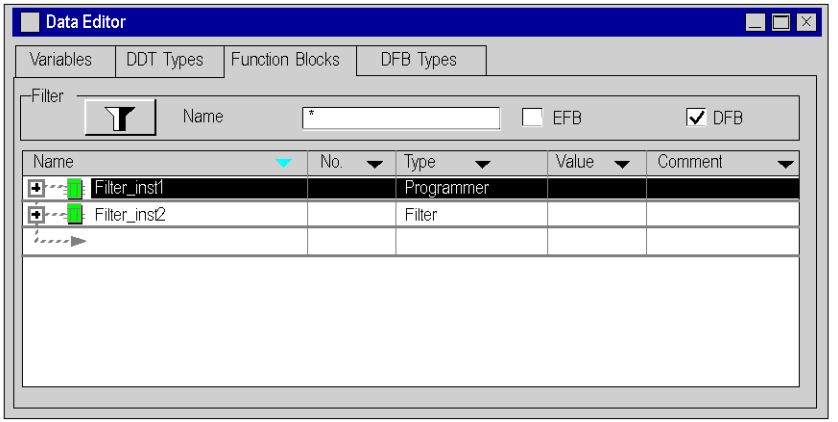
There are 2 different ways to modify the attributes of a DFB instance *(see Unity Pro, Program Languages and Structure, Reference Manual )*:

- From the **Function Blocks** register tab, for the attributes that are directly accessible,
- or via the **Data Properties** dialog box for all attributes.

## Modifying an attribute using the Function Blocks register tab

To modify the attributes of a DFB instance, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* register tab in the data editor. |
| 2 | Double-click on the cell that you wish to change the content of (you can also highlight this cell and press **Enter**).<br>**Result**: A selection list or an entry field is displayed.<br>  |

| Step | Action |
|---|---|
| 3 | Modify the attribute (instance name, instance type, comment) and confirm with **Enter**.<br>**Result**: The DFB instance is displayed with its new attribute.<br>  |

## Modifying an attribute via the Data Properties dialog box

To modify the attributes of a DFB instance using the **Data Properties** dialog box, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* register tab in the data editor. |
| 2 | Right-click on the DFB instance whose attributes you wish to view.<br>**Result**: A shortcut menu is opened. |
| 3 | Activate the **Properties** command.<br>**Result**: The related dialog box is opened. |
| 4 | In the **Value** column, double-click on the cell of the attribute to modify (you can also highlight this cell and press **Enter**). |
| 5 | Modify the attribute (instance name, instance type, comment) and confirm with **Enter**.<br>**Result**: The new attribute is assigned to the DFB instance. |

### Simultaneous modification of an attribute of several DFB instances

To simultaneously modify an attribute of several DFB instances, you must do the following:

| Step | Action |
|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* register tab in the data editor. |
| 2 | Select the DFB instances of which you want to change an attribute (adjacent or non-adjacent multi-selection). |
| 3 | Right-click on one of the instances that you just selected.<br>**Result**: A shortcut menu is opened. |
| 4 | Activate the **Properties** command.<br>**Result**: The related dialog box is opened. |
| 5 | Modify the attribute value and confirm with **Enter**.<br>**Result**: The new attribute is assigned to the DFB instance. |

### Limitations

The following limitations apply to multiple selections:

- All the selected instances must have the attribute and it must be modifiable,
- The new attribute value must be valid for all the selected instances,
- The attribute name cannot be changed (a DFB instance name is unique).

**NOTE:** In a multiple selection, the selected elements must at the same level of the hierarchy or must be compatible.

### Modifying the attributes of the elements of multiple DFB instances

Modification of one or several (multiple selection) attributes of elements of one or several DFB instances. To do this, you must:

| Step | Action |
|---|---|
| 1 | Open the **Function Blocks** *(see page 1273)* register tab in the data editor. |
| 2 | Open the DFB instance(s) to modify. |
| 3 | Select the elements for which you want to modify the attributes (by pressing **Shift** if the multiple selections are adjacent or **Ctrl** if the multiple selections are non-adjacent). |
| 4 | Right-click on one of the elements that you just selected.<br>**Result**: A shortcut menu is opened. |
| 5 | Activate the **Properties** command.<br>**Result**: The related dialog box is opened. |
| 6 | Modify the attribute value and confirm with **Enter**.<br>**Result**: The new attribute is assigned to the selected elements. |

# Authorized DFB Instance Modifications

## At a Glance

The modifications possible on a DFB instance *(see Unity Pro, Program Languages and Structure, Reference Manual )* are defined by the access rights authorized by your user profile *(see page 1269)*.

## Authorized DFB instance modifications

The following table shows the operations that can be carried out on the DFB instances in offline mode:

| Situation | Possible Actions |
|---|---|
| **Actions when there is no DFB instance selected** | Create a DFB instance, |
| | Paste a DFB instance. |
| **Actions when a DFB instance is selected** | Delete a DFB instance, |
| | Change the name of a DFB instance, |
| | Change the type of a DFB instance, |
| | Change the comment for a DFB instance, |
| | Duplicate a DFB instance. |
| **Actions when several DFB instances are selected** | Delete DFB instances, |
| | Change the type of the DFB instances, |
| | Change the comment for the DFB instances, |
| | Duplicate DFB instances. |
| **Actions on all DFB instances** | Export all DFB instances, |
| | Print locally all DFB instances. |

# Section 33.3
# How to Create Nested DFBs

## How to Create Nested DFBs

### At a Glance

When creating nested DFBs (DFBs with DFB instances as private variables), you must not exceed 8 nesting levels (including DDT variables).

You must also respect certain chronological rules. The following table describes the procedure.

### Procedure

The steps are as follows.

| Step | Action |
|---|---|
| 1 | Create the DFB type of the last level (**n**) |
| 2 | Create the DFB type of the level **n-1** |
| 3 | For this DFB type, create a private variable with a type matching the DFB type of level **n** |
| 4 | Create the DFB type of the level **n-2** |
| 5 | For this DFB type, create a private variable with a type matching the DFB type of level **n-1** |
| 6 | Repeat these actions **without exceeding 8 levels** |
| 7 | Create a **first** level DFB type instance, which you will use in your program. |

# Part V
# Project Configuration

## Subject of this Part

This part describes how to configure Premium/Atrium, Quantum and Modicon M340 PLC stations.

Its purpose is to guide the user in the configuration:

- of the PLC station,
- of the modules used in the station.

For Modicon M580 PLC stations, refer to chapter *Configuring the CPU in Unity Pro (see Modicon M580, Hardware, Reference Manual)*.

## What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 34 | Introduction to the configuration | 1287 |
| 35 | Bus Editors | 1299 |
| 36 | Modules editor | 1351 |
| 37 | General functionalities of the editor | 1389 |

# Chapter 34
# Introduction to the configuration

## Aim of this section

This section gives a very general introduction to the functionalities offered by the configuration editor and describes how to access the default configuration.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Proposed Services | 1288 |
| Services offered in offline mode | 1292 |
| Services Offered In Online Mode | 1293 |
| Access to project configuration | 1295 |

## Proposed Services

### Introduction

The configuration tool is used to:

- **create\modify\save** *(see Unity Pro, Program Languages and Structure, Reference Manual )* the elements involved in the configuration of the PLC station,
- **set up** the application-specific modules comprising the station,
- **diagnose** the modules configured in the station,
- **assess** the current discharged from the voltages supplied by the power supply module declared in the configuration,
- **control** the number of application-specific channels configured in relation to the capacities of the processor declared in the configuration.
- **assess** the processor memory *(see Unity Pro, Program Languages and Structure, Reference Manual )* usage.

The configuration may be performed before or after the programming of the project; this has the advantage of being able to create generic projects without having to be concerned with the configuration in the initial stage.

The various services are provided through 2 editors which are:

- the bus editor, which allows you to select and position the modules or devices on the bus,
- the input/output module editor, which allows you to set up the modules or devices present on the PLC station bus(es).

**NOTE:** When you configure the different elements of your project (specific application, modules, processors, etc.) you may ensure that there are no conflicts between data areas (overlap) as this may result in downgraded application functionality.

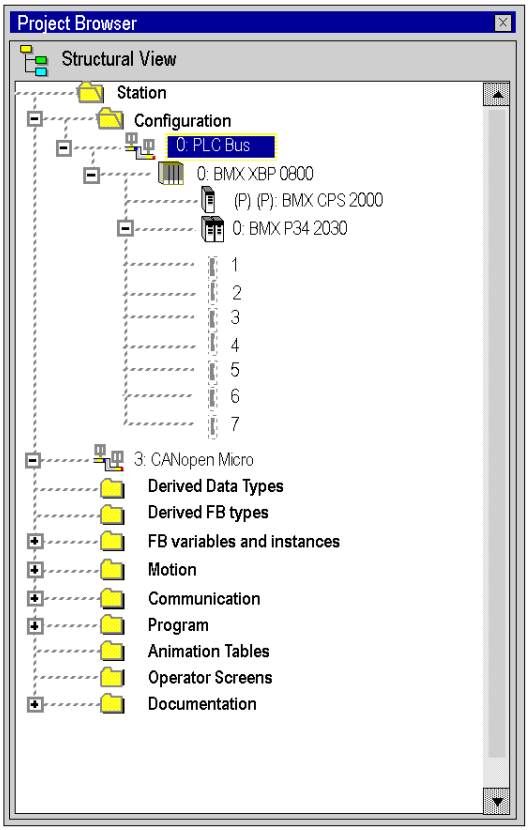
| ⚠ CAUTION |
| --- |
| **UNEXPECTED APPLICATION BEHAVIOR**<br>Before performing a build ensure there is no overlapping of data between specific applications within the same project. Check that the project operates correctly.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

## At a Glance

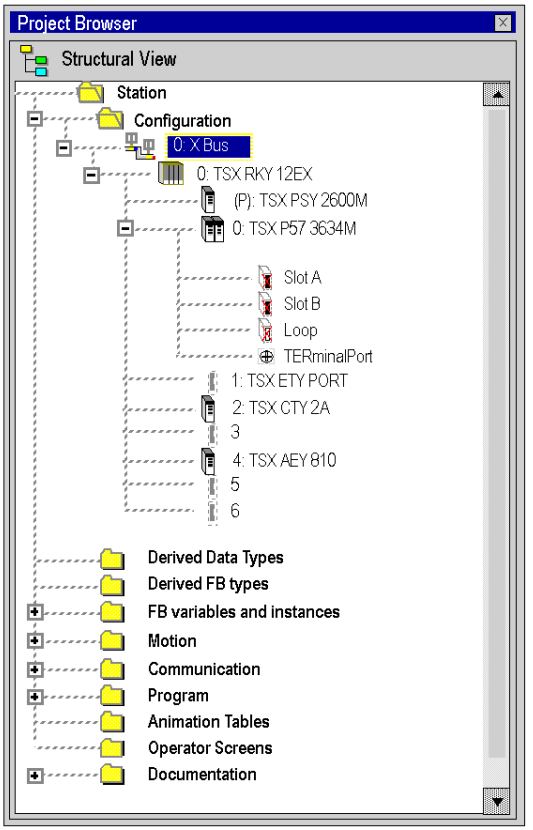
From the project browser, you can view the hardware configuration of the PLC station by opening the **Configuration** directory.

The hardware configuration of the station is displayed in a directory tree.

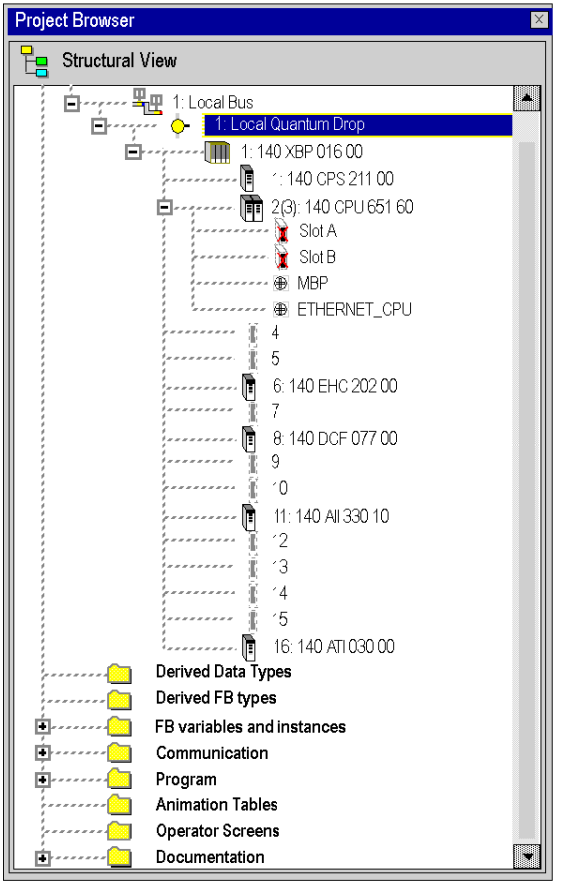
Configuration example for a Modicon M340 station:



Configuration example for a Premium station:



Configuration example for a Quantum station:




**NOTE:** you may note that each address (slot in the rack) is not necessarily occupied by an input/output module, and that some modules may occupy 2 slots.

# Services offered in offline mode

## Introduction

These functional options are preset by the bus editor and the input/output module editor, when the PLC is not connected to the programming terminal.

## From the bus editor

The available services are:

- selection/replacement of the rack(s),
- selection/replacement of the power supply module,
- selection/replacement/set-up of the processor,
- selection/positioning/replacement of an application-specific module or a device,
- selection/replacement of a PCMCIA card in a module or in the processor,
- the figures for:
  - the consumption of power provided by the power supply module,
  - the number of application-specific channels configured,
  - the memory usage of the PLC.

## From the input/output modules editor

The available services are:

- the set-up of the input/output modules,
- the set-up of certain PCMCIA cards,
- the set-up of the application-specific functions.

# Services Offered In Online Mode

## Introduction

These are configurable functional options when the PLC is connected to the programming terminal.

Certain services are available or not according to whether the type of PLC connected is:

- Modicon M340,
- Premium/Atrium,
- Quantum.

## Modicon M340 PLCs

Services offered by the configuration editor:

| Service | PLC in RUN mode | PLC in STOP mode |
|---|---|---|
| Adding/moving a module | No | No |
| Modification of configuration parameters | Yes | Yes |
| Modification of adjustment parameters | Yes | Yes |
| Display of errors | Yes | Yes |
| Display of status information | Yes | Yes |

## Premium\Atrium PLCs

Services offered by the configuration editor:

| Service | PLC in RUN mode | PLC in STOP mode |
|---|---|---|
| Adding/moving a module | No | No |
| Modification of configuration parameters | Yes | Yes |
| Modification of adjustment parameters | Yes | Yes |
| Display of errors | Yes | Yes |
| Display of status information | Yes | Yes |

## Quantum PLCs

All Quantum PLCs allow online modification. Services offered by the configuration are:

| Service | PLC in RUN mode | PLC in STOP mode |
|---|---|---|
| Automatic detection of the PLC configuration | Yes | Yes |
| Adding/removing a module | Yes | Yes |
| Modification of configuration parameters | Yes | Yes |
| Modification of adjustment parameters | Yes | Yes |
| Display of errors | Yes | Yes |
| Display of status information | Yes | Yes |

**NOTE:** All modifications made in the bus editor are authorized in online mode. But in order for these to be taken into account, you must generate and transfer the project to the PLC.

**NOTE:** A feature CCOTF modification *(see Modicon Quantum, Change Configuration On The Fly, User Guide)* has been developped for Unity Quantum and allows I/O configuration changes when the PLC is in RUN mode.

# Access to project configuration

## Introduction

When a project is created, a default configuration is created automatically according to the choices imposed by the programming software.

These choices concern:

- the PLC family,
- the type of processor used in the station.

These choices determine the power of the configuration, and the available functionalities (number of inputs/outputs, field buses, languages used, types of data, etc.).

**NOTE:** The choice of PLC family is irreversible, only the type of processor may be changed.

## Procedure

Carry out the following actions:

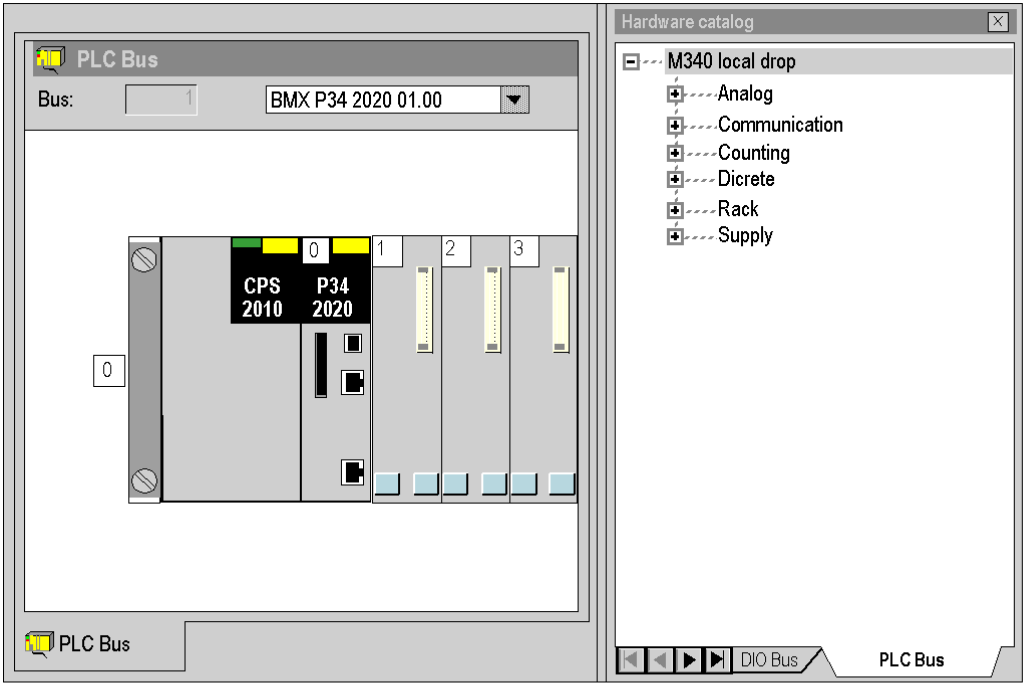
| Step | Action |
|---|---|
| 1 | From the project browser, open the **Configuration** directory. |
| 2 | **For a Modicon M340 station:**<br>From the **PLC Bus** directory or from the directory representing the station, via the contextual menu select the **Open** command; the default configuration is displayed on the screen.<br>**For a Premium station:**<br>From the **X-Bus** directory or from the directory representing the station, via the contextual menu select the **Open** command; the default configuration is displayed on the screen.<br>**For a Quantum station:**<br>From the directory representing the station, via the contextual menu select the **Open** command; the default configuration is displayed on the screen. |
| 3 | In the configuration window, via the contextual menu select the **Zoom in** command to determine the size of the station.<br>Click **Display** → **Zoom** → **Zoom to Fit** to optimize the size of the station in relation to the size of the window. |

At this stage the configured physical elements constituting the station are:

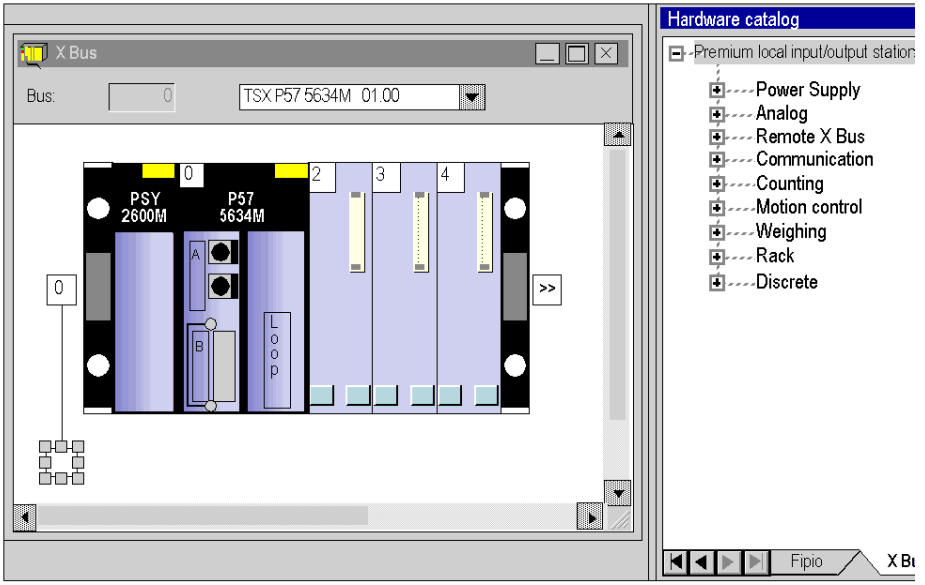
- the rack,
- the power supply module (except for a Quantum station),
- the processor.

## Configuration

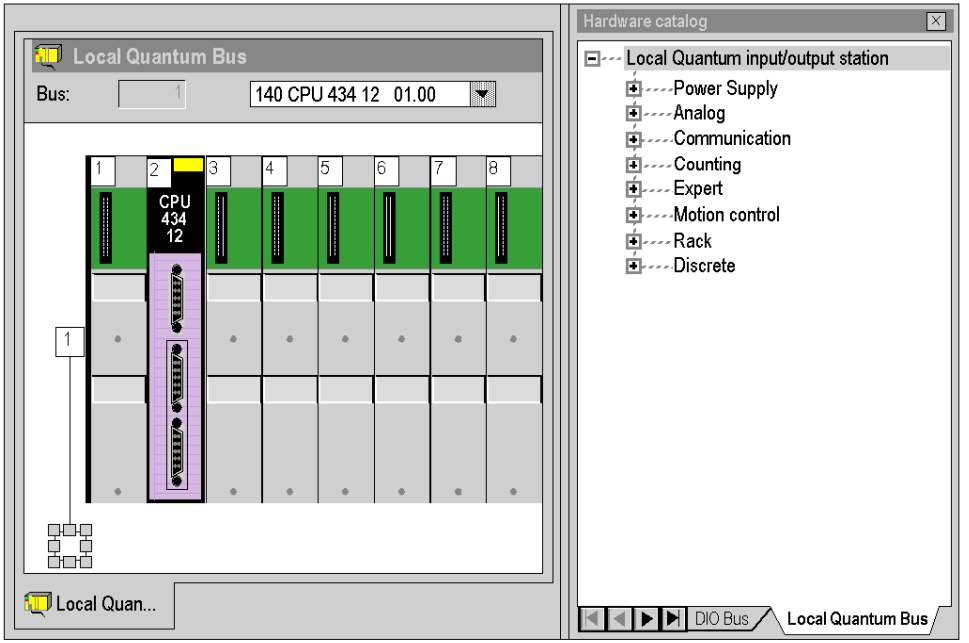
For a Modicon M340 station:



For a Premium station:



For a Quantum station:



# Chapter 35
# Bus Editors

## Aim of this section

This chapter guides the user in the creation of the physical configuration of the PLC station.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 35.1 | Rack configuration on local bus | 1300 |
| 35.2 | Configuration of the power supply modules | 1308 |
| 35.3 | Configuration of the processor | 1311 |
| 35.4 | Configuration of the modules in the PLC station | 1314 |
| 35.5 | Configuration of devices on the field bus | 1319 |
| 35.6 | Consumption management | 1340 |
| 35.7 | Bus editors in online mode | 1347 |

# Section 35.1
## Rack configuration on local bus

### Aim of this sub-section

This sub-section guides the user in the installation of the rack/s constituting the station.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Organization of the Racks | 1301 |
| How to configure the racks | 1306 |

# Organization of the Racks

## Introduction

A default rack is selected when the project is created. It has the following address:

- **0** for a PLC from the Premium/Atrium or Modicon M340 family,
- **1** for a PLC from the Quantum family.

This rack contains the type of processor selected when the project is created. This processor can be replaced by a compatible processor.

Processors belonging to the Modicon M340 family.

| Type of processors | Number of racks managed |
|---|---|
| For all M340 PLC Version 01.00. | 1 rack |
| For BMX P34 1000 Version 02.00 | 2 racks |
| For BMX P34 20X0 | 4 racks |

Processors belonging to the Premium/Atrium family

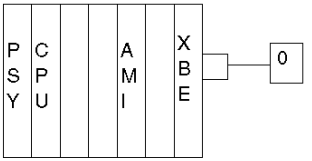
| Type of processors | Numbers of racks managed |
|---|---|
| TSX 57 0244 | 1 rack |
| TSX 57 1x4 | Up to 4 racks |
| TSX P57 204<br>TSX PCI 57 204<br>TSX P57 254<br>TSX P57 2634<br>TSX P57 304<br>TSX P57 354<br>TSX P57 3634<br>TSX P57 454 / TSX PCI 57 354<br>TSX P57 4634<br>TSX P57 554<br>TSX P57 5634<br>TSX P57 6634 | Up to 16 racks |

Processors belonging to the Quantum family

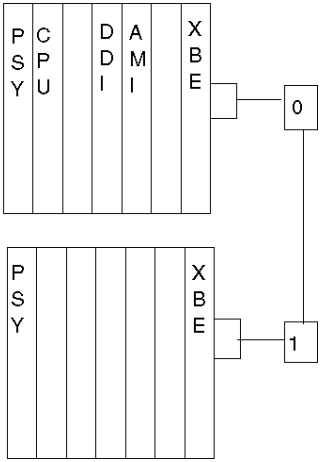
| Type of processors | Numbers of racks managed |
|---|---|
| 140 CPU 311-10 | Does not depend on the type of processor |
| 140 CPU 434-12A/U | |
| 140 CPU 534-14A/U | |
| 140 CPU 651-50\60\60S | |
| 140 CPU 652-60 | |
| 140 CPU 658-60 | |
| 140 CPU 671-60\60S | |
| 140 CPU 672-60\61 | |
| 140 CPU 678-61 | |

## Organization of a Modicon M340 Station on PLC Bus

Station composed of a single rack:

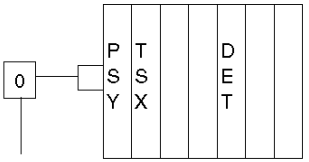


Station composed of several racks:

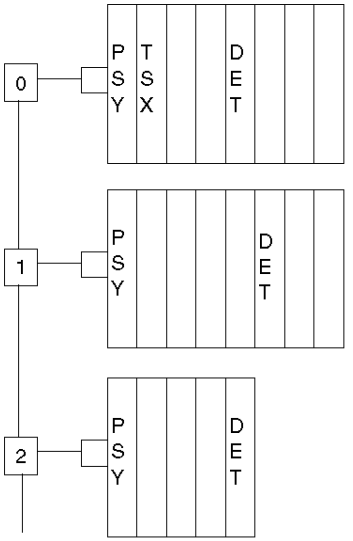



## Organization of The Premium/Atrium Station on Bus X

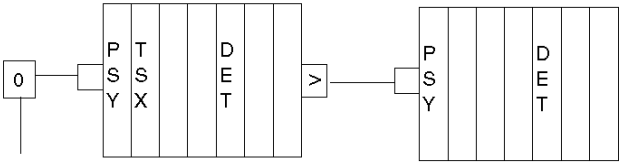
Station composed of a single rack (standard rack):



Station composed of several racks (extendable racks) with different addresses:

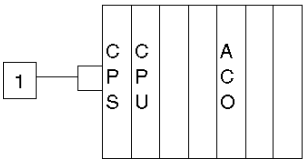



Station composed of several racks (extensible racks) with the same address:

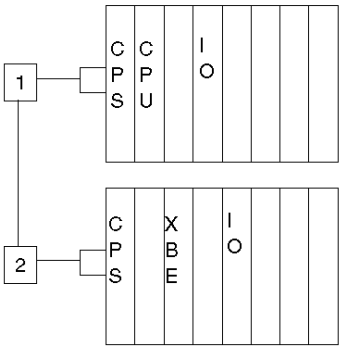


## Organization of a Quantum Station on Local Bus

Station composed of several racks:




Station composed of several racks (extendable racks) with different addresses:



# How to configure the racks

## Introduction

You may manipulate the racks of a PLC station:

- either by using the functions provided by the bus editor,
- or from the browser provided by the bus editor.

## Functions offered by the bus editor

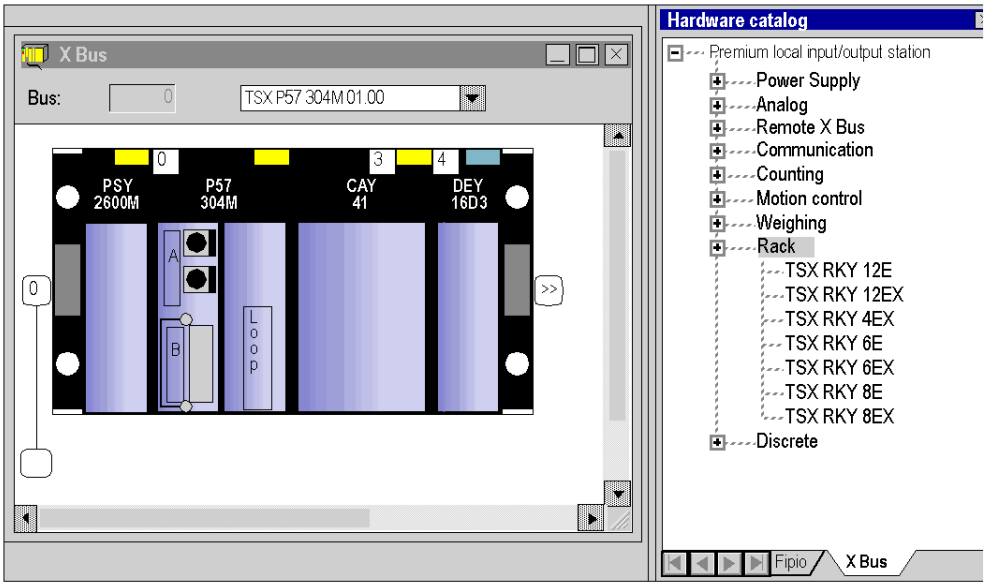
From the bus editor the following functionalites are offered:

| If you want to | then | and |
|---|---|---|
| **select** a rack | select the address of the rack  eight handles appear around the selected rack. | |
| **Cut/Paste** a rack | select the rack and, from the contextual menu, click **Cut** | select the destination address of the rack and, using the contextual menu, click **Paste**. |
| **Copy/Paste** a rack | select the rack and, using the contextual menu, click **Copy** | select the destination address of the rack and, using the contextual menu, click **Paste**. |
| **add** a rack | select an empty address  or the extension symbol  | using the contextual menu, click **New Device**, |
| **replace** a rack | select the rack and, using the contextual menu, click **Replace Rack**, | select the required rack from the list offered. |
| **delete** a rack | select the rack | using the contextual menu, click **Delete Rack**, |
| **empty** a rack | select the rack | using the contextual menu, click **Clear Rack**, |

## Bus editor browser

The browser only allows for the addition of one rack in the PLC station.

Example for a Premium station:




Carry out the following actions:

| Step | Action |
|---|---|
| 1 | From the browser, select the **Rack** directory and open it by clicking on **+** |
| 2 | select the required rack and, holding down right mouse button, move it to the required address point; a dialog box is displayed. |
| 3 | Specify in the dialog box the required address and confirm with **OK**. |

# Section 35.2
# Configuration of the power supply modules

## How to configure the power supply modules

### Introduction

When creating an application two boxes are produced according to whether you select a **Premium/Atrium** station, a **Quantum** station or a **Modicon M340** station:

- in a Modicon M340 or Premium station a power supply module is configured by default,
- in a Quantum station, there is no power supply module configured by default.

### Rules for a Modicon M340 station

The power supply module must occupy the left-most position of the rack. This position does not have an address.

**There is only one power supply module per rack**.

### Rules for a Premium/Atrium station

The power supply module must occupy the left-most position of the rack. This position does not have an address.

A double format power supply module also occupies the address position 0 (usually occupied by the processor module), in this case the processor module must be configured at the address position 1.

**There is only one power supply module per rack**.

### Rules for a Quantum station

The power supply module can occupy any rack position. It has an address.

The power supply modules are single format.

**Several power supply modules may be configured in a rack**.

## Functions offered by the bus editor

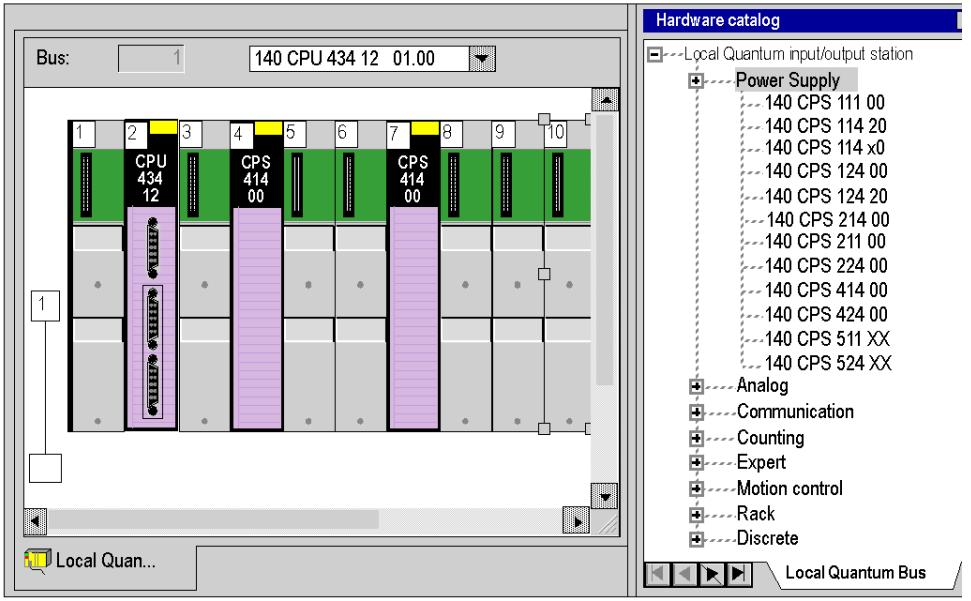
From the bus editor the following functionalites are offered:

| If you want to | then | and |
|---|---|---|
| **Select** a module | click on it. Eight handles appear around the module. | |
| **Copy/Paste** the module | select the module and, using the contextual menu, click **Copy** | select the target position and, using the contextual menu, click **Paste**. |
| **Cut/Paste** the module | select the module and, using the contextual menu, click **Cut** | select the target position and, using the contextual menu, click **Paste**. |
| **Add** a module | select the position in the rack required and, using the contextual menu, click **New Device** | select from the proposed list the module required. |
| **Move** a module | select the module | holding down the left mouse button, move it to the required position. |
| **Delete** a module | select the module | using the contextual menu, click **Delete Module**. |

## Bus editor browser

The browser is used to put or to add (for Quantum) a power supply module in the station.

Example of a Quantum station containing 2 power supply modules:




Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the browser, select the directory **Supply**, and display it by clicking on **+**. |
| 2 | Select the power supply module required and, holding down the left mouse button, move it to the required position. |

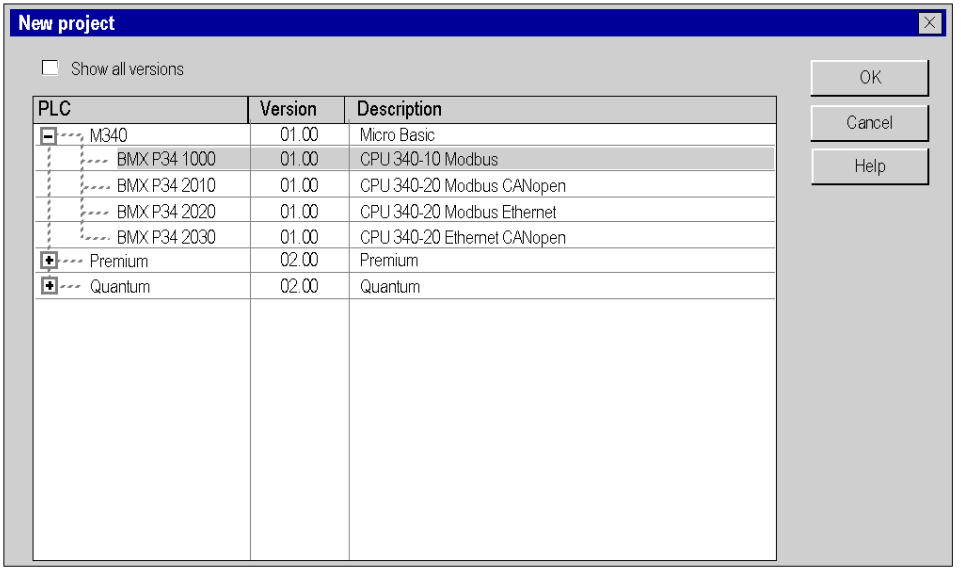
# Section 35.3
# Configuration of the processor

## Choice and replacement of the processor

### Selecting the processor

The choice of processor is made when the project is created, this choice is not irreversible.

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | From the welcome screen of the software, select the **New** command from the **File** menu. |
| 2 | Open the type of PLC required.<br>Example:<br>  |
| 3 | If you want to see all PLC versions, click **Show all versions**. |
| 2 | Select the processor. |
| 4 | Validate with **OK**. |

### Rules

For a Modicon M580 station:
- The processor occupies the slots 00 and 01.
- The processor is double format.

For a Modicon M340 station:
- The processor occupies the slot 0.
- The processor is single format.

For a Modicon Premium station:
- The processor occupies the slot 0, it may occupy slot 1 if a double format power supply is configured.
- The processor is single format or double format (in this case it occupies two slots).

For a Modicon Quantum station:
- The processor can occupy any position in the rack.
- The processor is single format or double format (in this case it occupies two slots).

You cannot move or copy a processor in another rack. The processor cannot be deleted from the rack, it can only be replaced.

### Modicon M580 processor restriction

The following Modicon M580 processor version cannot be replaced:
- Upgrade: OS version < V2.00 cannot be replaced by an OS version ≥ V2.00.
- Downgrade: OS version ≥ V2.00 cannot be replaced by an OS version < V2.00.

If a processor replacement mentioned previously is needed, proceed as follows:

| Step | Action |
|---|---|
| 1 | Export *(see page 1719)* the original application program and/or elements (I/O configuration, DFB type, animation table, ...). |
| 2 | Create a new configuration with the new Modicon M580 processor OS version. |
| 3 | Import *(see page 1719)* original application program and/or elements. |

### Replacement of the processor

The configuration editor helps you if you wish to replace the processor. A message is sent if a replacement is not authorized.

The new processor has to belong to the same PLC family as the previously configured processor.

If certain previously configured input/output modules are no longer supported by the new processor, when the project is analyzed, error messages are displayed. You have to resolve these incompatibilities.

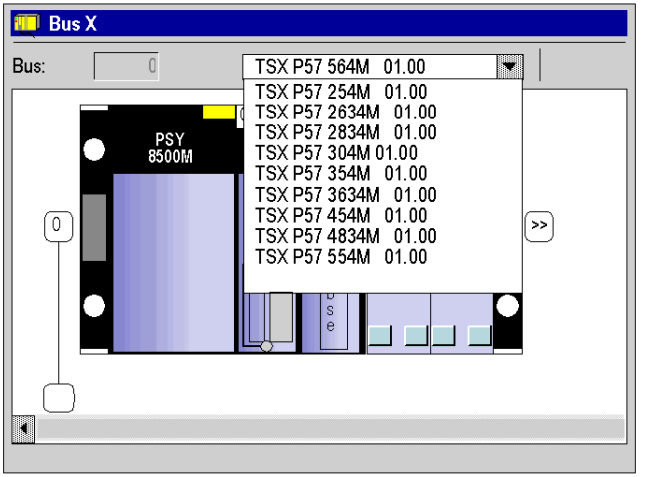
**NOTE:** This operation is only possible in offline mode (PLC not connected).

You may proceed in accordance with two methods:

Method 1:

| Step | Action |
|---|---|
| 1 | From the bus editor, select the processor. |
| 2 | Using the contextual menu, click **Replace Processor...**. |
| 3 | Select the required processor. |
| 4 | Validate with **OK**. |

Method 2:

| Step | Action |
|---|---|
| 1 | From the bus editor, display the list of processors. To see all the processor versions, select **Show all versions**.<br>Example:<br>  |
| 2 | Select the required processor. |
| 3 | Confirm your selection. |

**NOTE:** When making a processor replacement, if an FFB used in the application is not available for the selected processor, you have to remove it from the Application sections and you have to remove it from the Application itself by the **Tools** → **Types Library Manager** → **Purge Unused Types** option.

# Section 35.4
# Configuration of the modules in the PLC station

## How to configure the modules in the PLC station

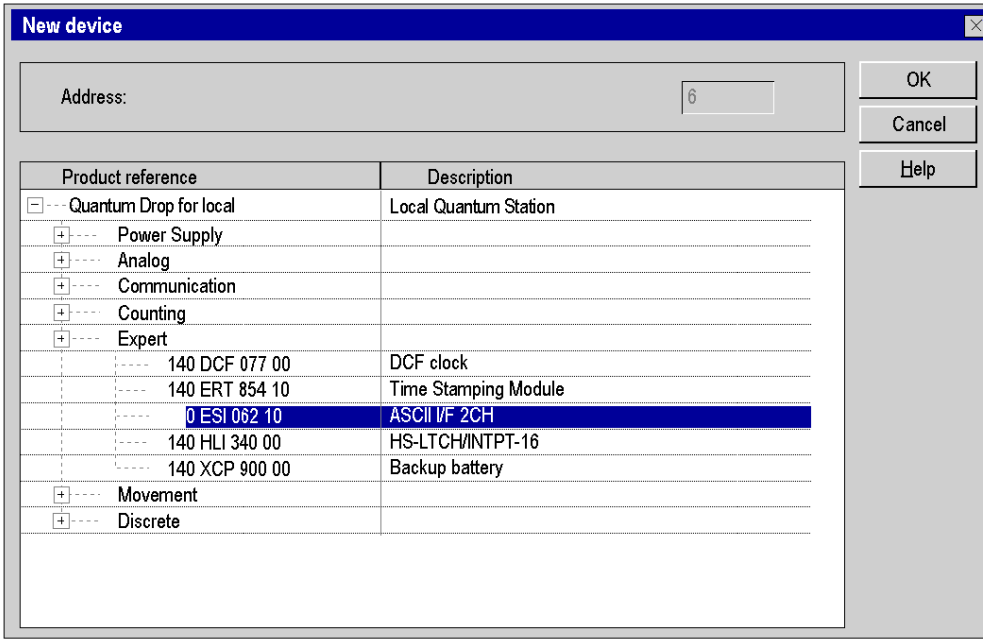
### Introduction

You have two options for inserting the modules in the rack:

- either you use the functions provided by the bus editor,
- or you use the hardware catalog provided by the bus editor.

## Positioning a module

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Use the mouse to select the position of the module to insert; eight handles appear around the module. |
| 2 | Using the contextual menu, click **New Device**. |
| 3 | Select the required application and module.<br>Example:<br>  |
| 4 | Click **OK** to confirm. |

**NOTE:** The double format modules are shown on the list of modules available when the slot preceding or following the selected slot is vacant.
In the case of a Premium or a Modicon M340, the position 0 may only be occupied by a double format power supply or a processor.

## Moving a module

A module may be moved:

- into the rack,
- into another rack if the station has several of them.

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | With the mouse, select the required module. |
| 2 | Drag and drop the module into new position. |

**NOTE:** If the module cannot be moved, this is indicated by the bus editor.

**Rules:**

The objects associated with the module are:

- deleted, then re-created automatically at the new address,
- replaced in:
  - the program,
  - the variable editor after user confirmation,
  - the animation tables,
  - the operator screens.

The **symbols** associated with the moved module objects are attached to the objects at the new address.

The module that has been moved keeps all its parameters.

For a **Modicon M340**,**Premium** or **Atrium** station, if a discrete module with a **RUN/STOP**-configured channel is moved, the **RUN/STOP** bit address remains unchanged. It is necessary to ensure that the **RUN/STOP** input corresponds to a valid discrete input address.

When the addresses of the module are used in an EF (e.g.: Send_REQ, Read_VAR,...), these are not updated automatically.

## Copy a module

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | With the mouse, select the required module. |
| 2 | Using the contextual menu, click **Copy**. |
| 3 | Use the mouse to select the target position, then using the contextual menu click **Paste**. |

**Rules:**

The objects associated with the module are copied to the new address.

The duplicated module keeps all its parameters.

The duplication of a module associated with an event (at least one module channel is associated with an event) is not possible. An event may be associated with several **channels/modules** at the same time.

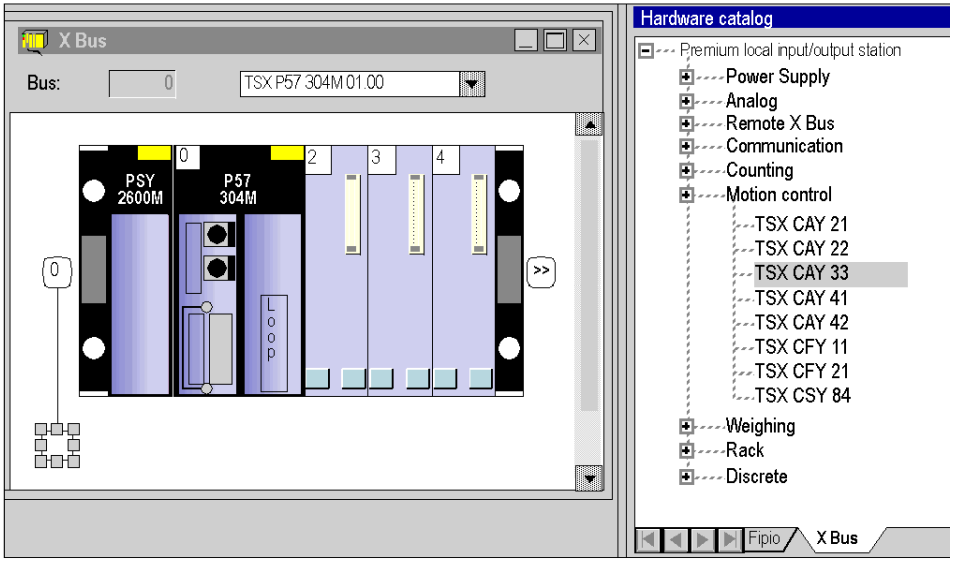
## Delete a module

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | With the mouse, select the required module. |
| 2 | Via the popup menu the select the **Delete Module** command. |

## Hardware catalog

Using the Hardware catalog, you can insert modules in the rack:




Carry out the following actions:

| Step | Action |
|---|---|
| 1 | From the **Configuration** directory in the project browser, select the subdirectory representing the station.<br>**Result**: the bus editor and hardware catalog appear. |
| 2 | Select the directory of the required specific application in the hardware catalog, and expand it by clicking on **+**. |
| 3 | Select the module, and, holding down the mouse button, move it to the required position on the rack in the bus editor. |

# Section 35.5
## Configuration of devices on the field bus

### Aim of this sub-section

This sub-section guides the user in the management of devices on field buses managed by the Modicon M340, Premium and Quantum stations.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Configuration of the CANopen Slaves for Modicon M340 | 1320 |
| How to access and create a Fipio field bus | 1324 |
| Creating and Accessing RIO\DIO Field Buses | 1328 |
| How to configure devices on the RIO/DIO bus | 1334 |

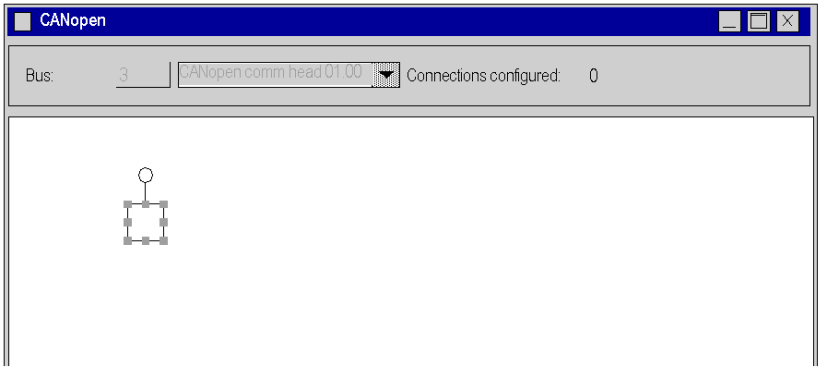
# Configuration of the CANopen Slaves for Modicon M340

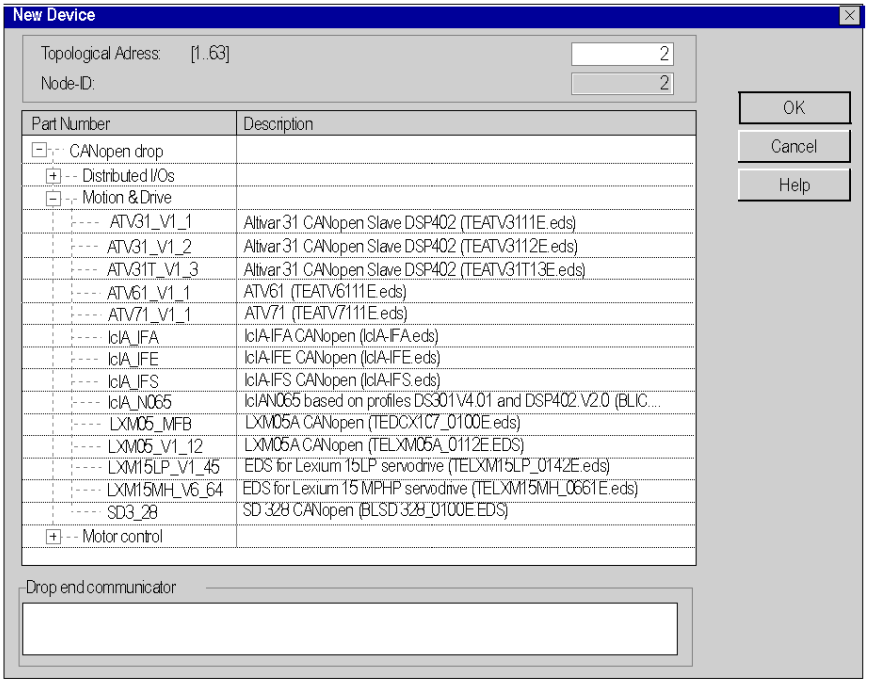
## Introduction

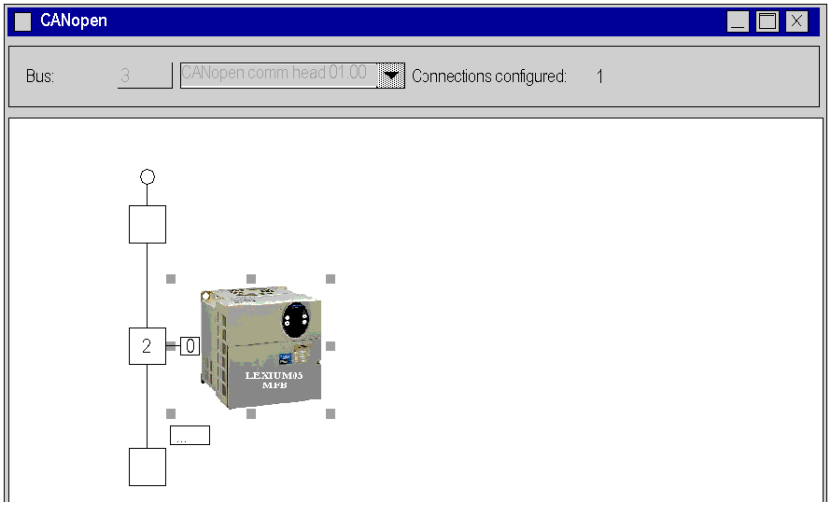
For Modicon M340 PLCs you can configure the CANopen bus with slaves (63 slaves maximum). The configuration is completely carried out with Unity Pro, the use of another software is not necessary.

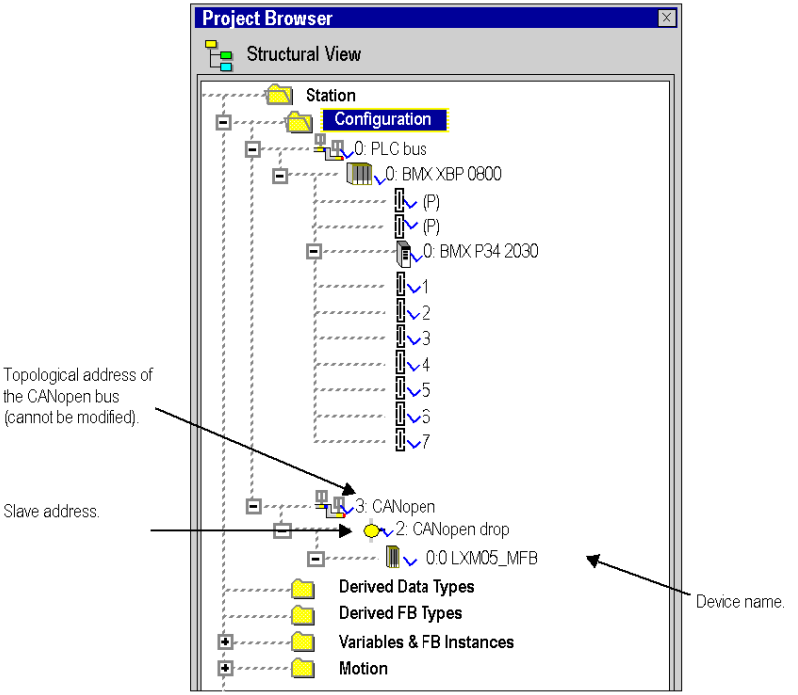
## How to Configure the CANopen Slave

This table describes the procedure to configure the CANopen slave.

| Step | Action |
|---|---|
| 1 | In the Unity Pro **Project Browser**, fully expand the **Configuration** directory and then double-click on **CANopen** .<br>**Result:** The CANopen window appears:<br>  |

| Step | Action |
|---|---|
| 2 | Select **Edit → New device**.<br>**Result:** The New Device window appears:<br>  |
| 3 | Set the address slave in Topological Address.<br>Choose the slave device. |

| Step | Action |
|---|---|
| 4 | Click on OK to confirm the choice.<br>**Result:** The CANopen window appears with the new device selected:<br>  |

| Step | Action |
|---|---|
| 5 | On the project browser the CANopen bus appears with the slaves.   |

## Other functions

You can also add, delete, move *(see Modicon M340 with Unity Pro, CANopen, User Manual)* a slave on the bus.

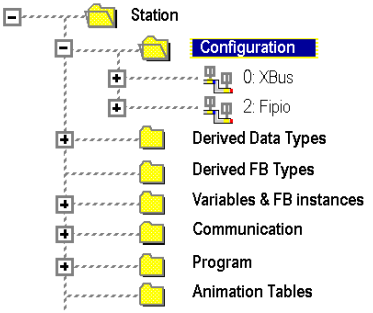
# How to access and create a Fipio field bus

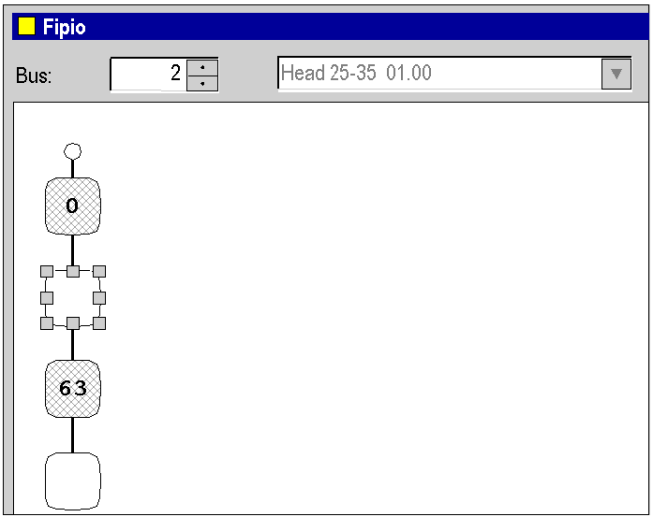
## Introduction

**Premium** PLCs offer a decentralized input/output architecture solution with the **Fipio** bus. You can configure *(see Premium and Atrium Using Unity Pro, Fipio Bus, Setup Manual)* 127 devices.

## Procedure

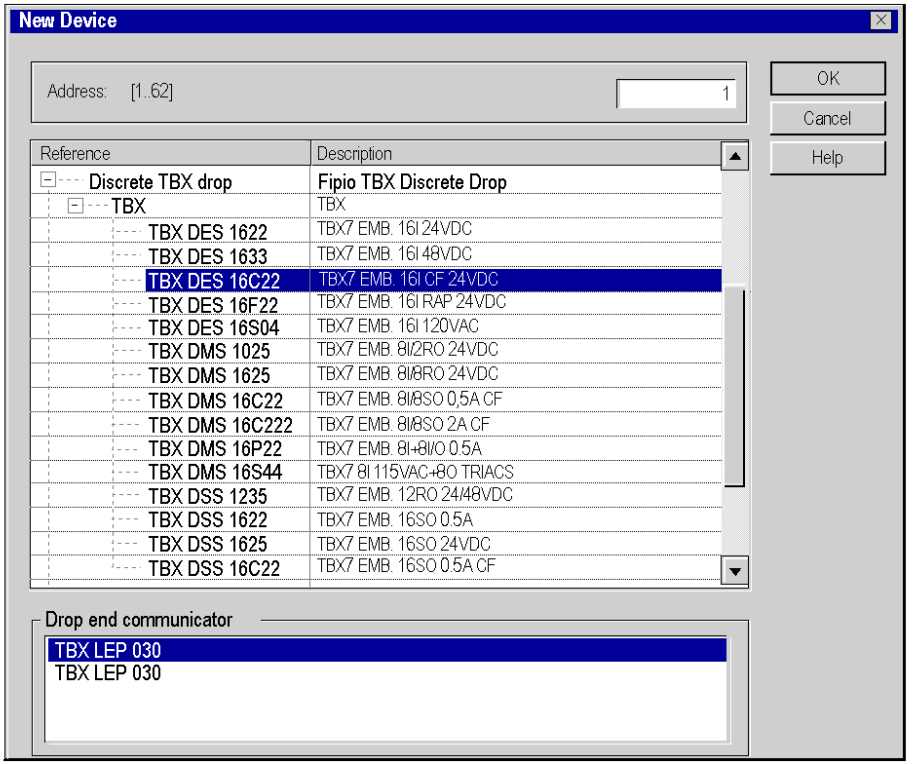
To access the Fipio field bus, perform the following actions:

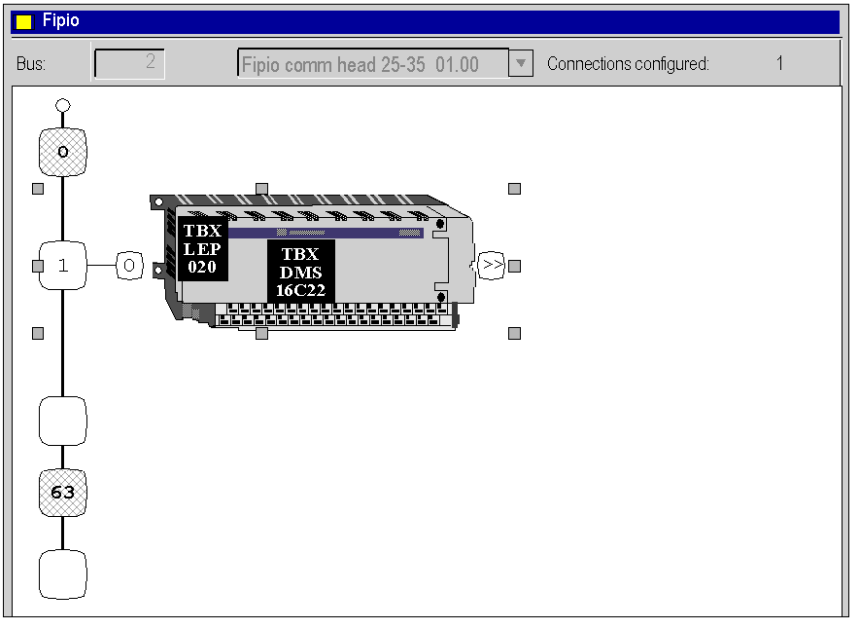
| Step | Action |
|---|---|
| 1 | From the project navigator, deploy the **Configuration** directory.<br>**Result**: the following screen appears<br>  |

| Step | Action |
|---|---|
| 2 | Select the Fipio sub-directory and select the **Open**command using the contextual menu.<br>**Result**: the **Fipio** window appears<br>  |

## Creating a Fipio bus

This operation is used to add, via the software, a device connected to the Fipio bus.

| Step | Action |
|---|---|
| 1 | Access the Fipio configuration screen: |
| 2 | Right-click on the logical address of the connection point, at the location where the module has to be connected (available addresses from 1 to 62 and 64 to 127, addresses 0 and 63 being reserved by the system).<br>**Result**: the **New Device** screen appears.<br>  |
| 3 | Enter the number of the connection point corresponding to the address.<br>By default, the Unity Pro software offers the first free consecutive address. |
| 4 | In the **Reference** field, enter the type of device to connect to the bus. |
| 5 | In the **Communicator** field, select the element type enabling communication on the Fipio bus.<br>For modules with built-in communicators, this window does not appear. |

| Step | Action |
|---|---|
| 6 | Confirm with **Ok**.<br>**Result:** the module is declared.<br>  |

## Other functions

You can also:

- add an extension module *(see Premium and Atrium Using Unity Pro, Fipio Bus, Setup Manual)* to the bus,
- delete, copy, paste *(see Premium and Atrium Using Unity Pro, Fipio Bus, Setup Manual)* bus devices,
- change *(see Premium and Atrium Using Unity Pro, Fipio Bus, Setup Manual)* the Fipio communicator.

# Creating and Accessing RIO\DIO Field Buses

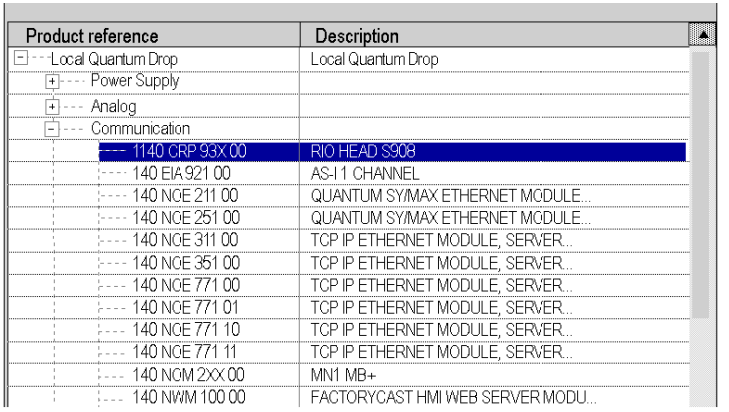
## Introduction

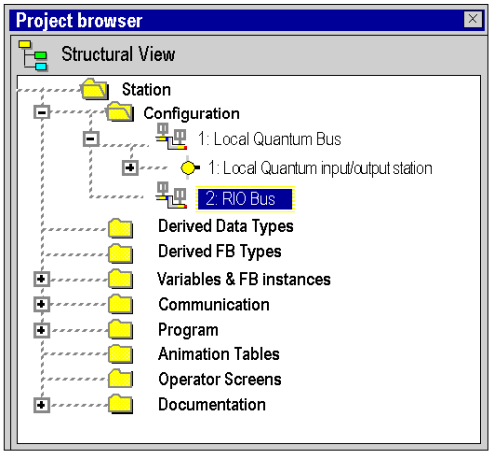
Quantum PLCs offer a decentralized input/output architecture solution:

- **RIO** field bus networks are based on the S908 input/output decentralization network technology. Up to 31 decentralized stations may be configured, with each station capable of supporting up to 128 input/output words.
- **DIO** field bus networks are based on Modbus Plus technology. 32 subscribers may be configured over 500 meters/1640 feet (receiving 64 subscribers over 2000 meters/6560 feet).

## Creating a RIO Bus

The following table describes the procedure for creating a RIO bus from a communication module:

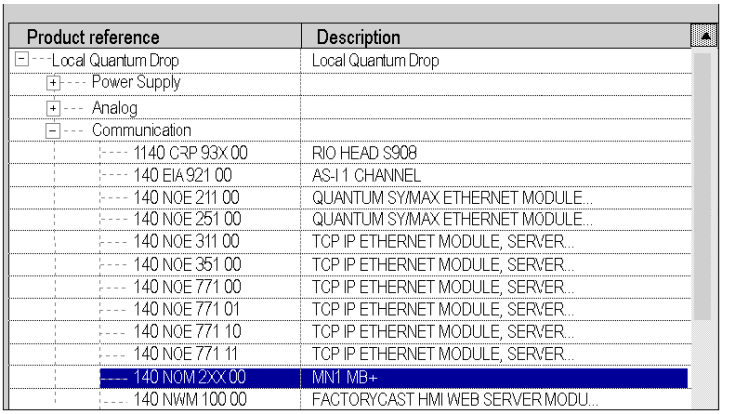
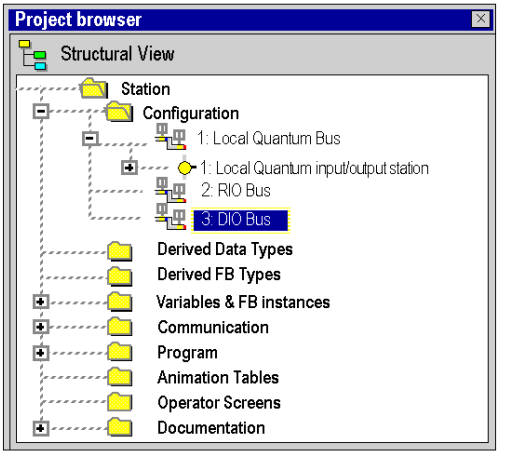
| Step | Action |
|---|---|
| 1 | In the bus editor, select the slot where you wish to insert the communication module. |
| 2 | Select **New Device** in the contextual menu.<br>**Result:** The **New Device** window appears. |
| 3 | Expand the *Communication* directory.<br>**Result:** The following window appears:<br>  |

| Step | Action |
|---|---|
| 4 | To create a RIO bus, select a 140 CRP 93x 00 module.<br>**Result:** The bus appears in the project browser:<br>  |

## Creating a DIO Bus

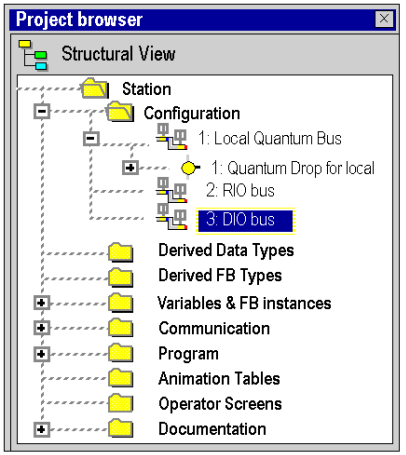
The following table describes the procedure for creating a DIO bus from a communication module:

| Step | Action |
|---|---|
| 1 | In the bus editor, select the slot where you wish to insert the communication module. |
| 2 | Select **New Device** in the contextual menu.<br>**Result:** The **New Device** window appears. |

| Step | Action |
|---|---|
| 3 | Expand the *Communication* directory.<br>**Result:** The following window appears:<br>  <br> |
| 4 | To create a bus, select a 140 NOM 2XX 00 module.<br>**Result:** The module appears in the rack. |
| 5 | Double-click the 140 NOM 2XX 00 module's Modbus Plus port.<br>**Result:** The bus configuration window appears. |
| 6 | Check the box marked **DIO Bus**. |
| 7 | Confirm the configuration.<br>**Result:** The DIO bus appears in the project browser.<br>  <br> |

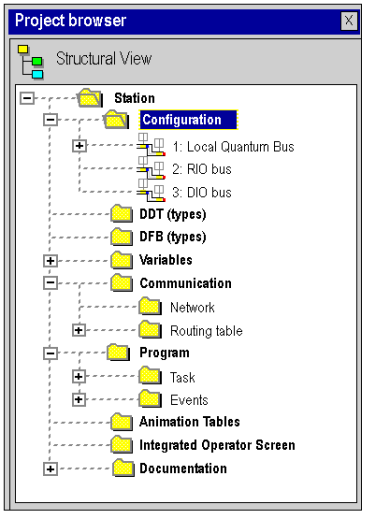
## Creating a DIO Bus from the Processor

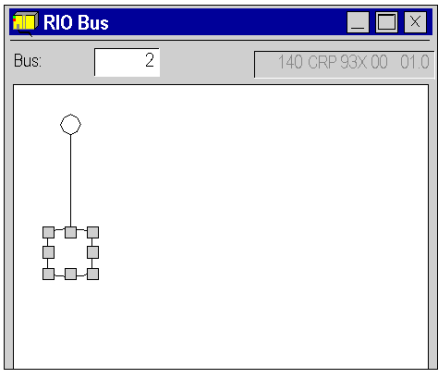
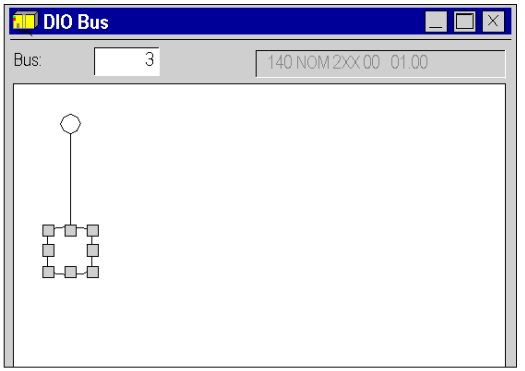
The following table describes the procedure for creating a DIO bus from the processor:

| Step | Action |
|---|---|
| 1 | In the bus editor, double-click the processor's Modbus Plus port. |
| 2 | Check the box marked **DIO Bus**. |
| 3 | Confirm the configuration.<br>**Result:** The DIO bus appears in the project browser:<br>  |

## Accessing a RIO or DIO Bus

To access a bus, carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the project browser, open the *Configuration* directory.<br>Example:<br>  |

| Step | Action |
|---|---|
| 2 | Select the *RIO bus* or *DIO bus* subdirectory depending on the type of bus you want to open. Next, select the **Open** command in the popup menu.<br>**Result:** The following window is displayed for the RIO bus:<br>  <br>**Result:** The following window is displayed for the DIO bus:<br>  |

# How to configure devices on the RIO/DIO bus

## Introduction

You can configure devices on field buses using the functions provided by the bus editor.

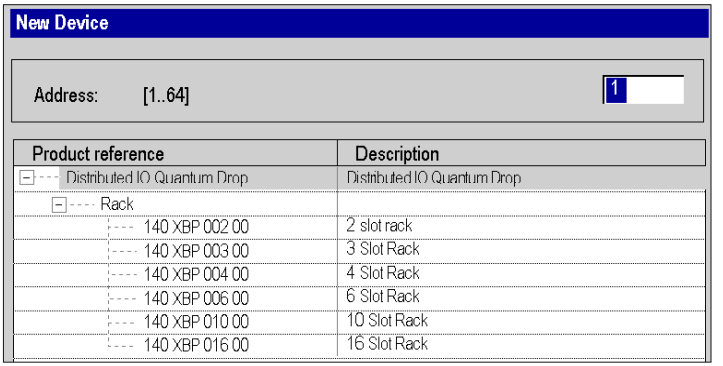
You can configure modules in the devices on field buses according to two methods:

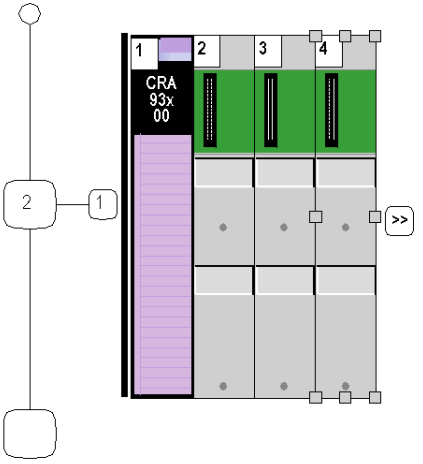
- either by using the functions provided by the bus editor,
- or from the hardware catalog provided by the bus editor.

You can also make copies, perform movements, or delete modules from a field bus device.

## Configuration of a device

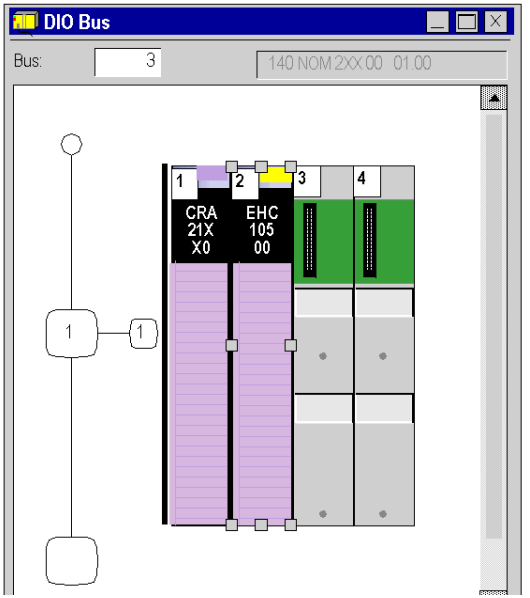
Carry out the following actions:

| Step | Action |
|---|---|
| 1 | From the **Configuration** directory in the project browser, select the **RIO Bus** or **DIO Bus** subdirectory. |
| 2 | Select **Open** in the contextual menu. |
| 3 | Select the connection point and via the popup menu select the **New Device** command.<br>**Result**: The **New Device** window appears.<br>  |

| Step | Action |
|---|---|
| 4 | Select the required rack and validate with **OK**.<br>**Result**: The rack appears in the bus.<br>  |

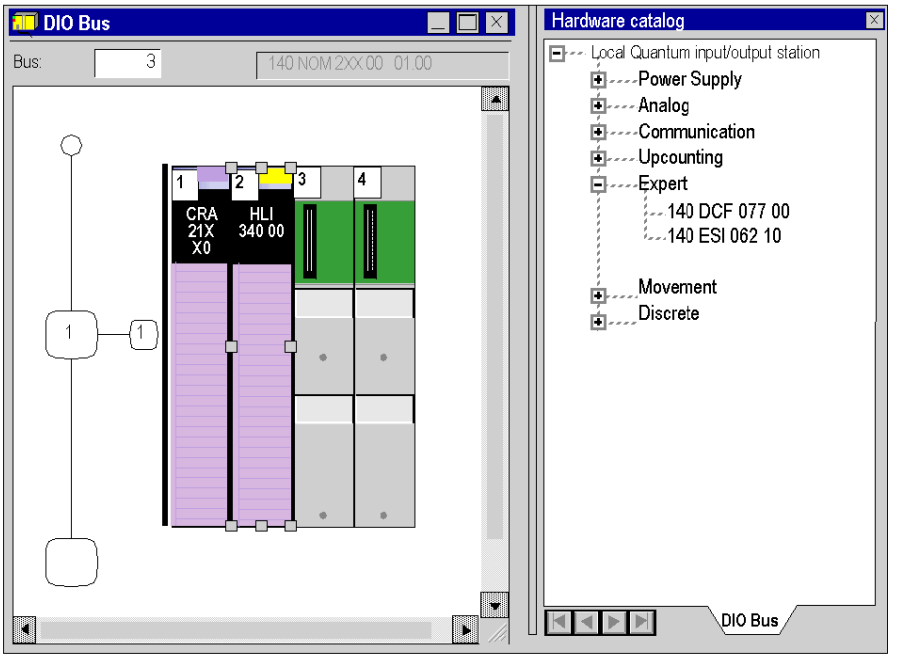
## Configuration of a module in a device

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the required rack (see procedure above: Configuration of a device). |
| 2 | To configure a module in the rack, select the module slot. |
| 3 | Select **New Device** in the contextual menu.<br>**Result**: The **New Device** window appears. |
| 4 | Select the module to be inserted.<br>**Result**: The module appears in the rack.<br>  |

## Hardware catalog

Using the hardware catalog, you can insert modules into a device on the field bus:




Proceed with the following steps:

| Step | Action |
|---|---|
| 1 | From the **Configuration** directory in the project browser, select the **DIO Bus** subdirectory.<br>**Result**: The field bus editor and hardware catalog appear. |
| 2 | Select the directory of the required specific application in the hardware catalog, and expand it by clicking on **+**. |
| 3 | Select the module, and, holding down the mouse button, move it to the required position on the rack in the bus editor. |

## Copy a module

Proceed with the following steps:

| Step | Action |
|---|---|
| 1 | With the mouse, select the required module. |
| 2 | Using the contextual menu, click **Copy**. |
| 3 | With the mouse, select the target position, then via the popup menu select the **Paste** command |

**Rules:**

The duplicated module keeps all its parameters.

The duplication of a module associated with an event (at least one module channel is associated with an event) is not possible. An event may be associated with several **channels/modules** at the same time.

## Moving a module

A module may be moved:

- into the rack,
- into another rack if the station has several of them.

Proceed with the following steps:

| Step | Action |
|---|---|
| 1 | With the mouse, select the required module. |
| 2 | Drag and drop the module into new position. |

**NOTE:** If the module cannot be moved, this is indicated by the bus editor.

**Rules:**

The objects associated with the module are:

- deleted, then re-created automatically at the new address,
- replaced in:
  - the program,
  - the variable editor after user confirmation,
  - the animation tables,
  - The operator screens.

The **symbols** associated with the moved module objects are attached to the objects at the new address.

The module that has been moved keeps all its parameters.

## Deleting a module

Proceed with the following steps:

| Step | Action |
|---|---|
| 1 | With the mouse, select the required module. |
| 2 | Via the popup menu, select the **Delete Module** command. |

# Section 35.6
# Consumption management

## Aim of this sub-section

This sub-section guides the user during the selection of modules so as not to exceed the limits:

- for consuming power,
- for consuming application-specific channels.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Power consumption budget | 1341 |
| Budget on the number of application-specific channels | 1344 |

# Power consumption budget

## At a Glance

A power consumption budget is made for:
- the rack power supply module,
- each module (processor, input/output module) depending on the rack power supply module.

This budget is presented in the form of a bar chart where each color has a particular significance. For each voltage, it indicates:
- the current power flow rate: **green**,
- the quantity of power still available: **white**,
- a power overload: **red**, when an excess occurs, a message is displayed.
- the total power (same color code).

## Power supply module budget

This consumption budget shows the amount of power discharged **by the power supply** for each voltage it supplies, as well as the total power.

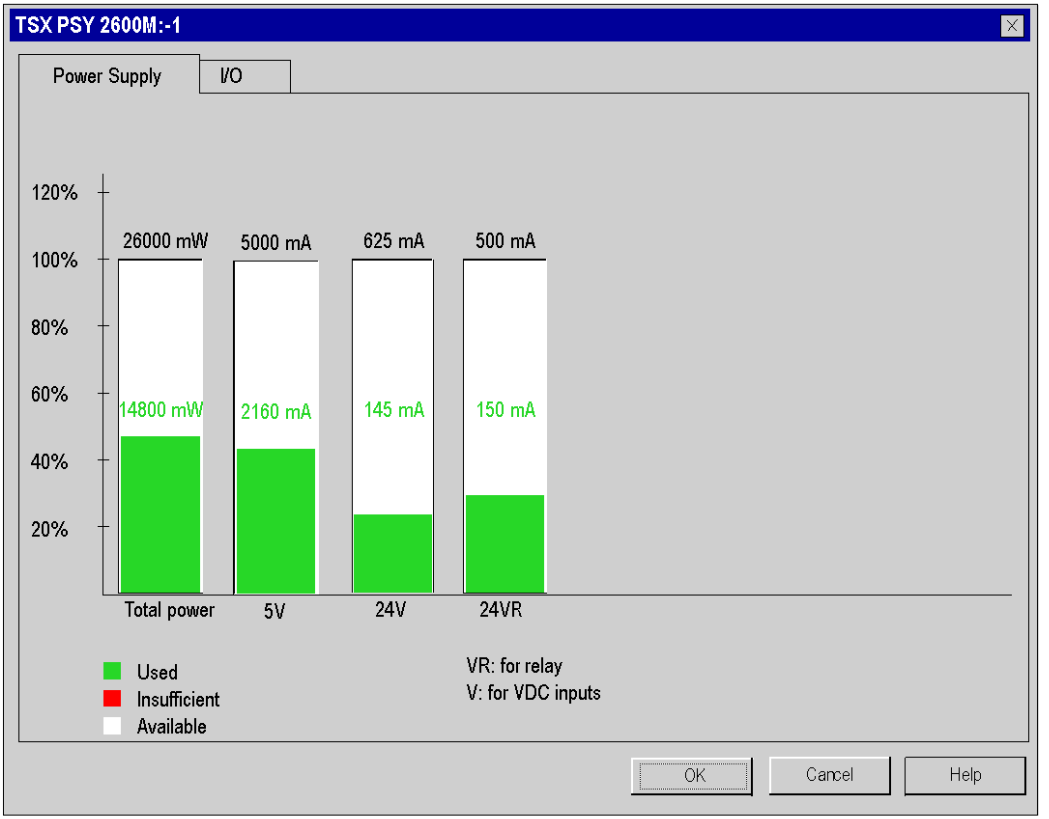
When you add or remove a module, the budget is adjusted on opening the **Power Supply and IO Budget** window.

**NOTE:** The Power Supply and IO Budget window must be closed to be able to delete or add a module.

How to access a consumption budget:

| Steps | Action |
|---|---|
| 1 | With the mouse, select the power supply module. |
| 2 | Using the contextual menu, click **Power Supply and IO Budget**. The bar chart is displayed. |
| 3 | Select the **Power Supply** tab. |

Budget:




**NOTE:** The ASY 800 24 V power consumption is always included in 24 VR (Internal Rack Power) even if configured with an external power supply.

## Budget of the other modules

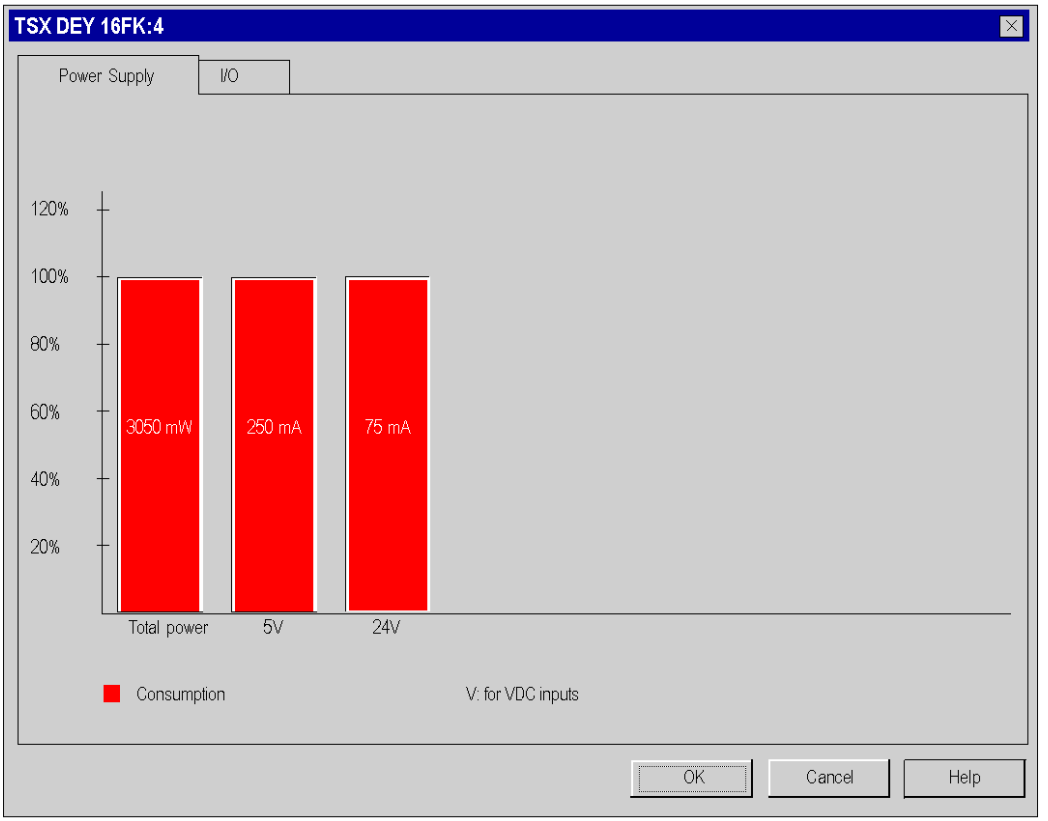
This consumption budget shows the amount of power discharged in **the module** for each voltage it uses, as well as the total power.

**NOTE:** The Power Supply and IO Budget window must be closed to be able to delete or add a module.

How to access a consumption budget:

| Steps | Action |
|---|---|
| 1 | With the mouse, select the module of your choice. |
| 2 | Using the contextual menu, click **Power Supply and IO Budget**. The bar chart is displayed. |
| 3 | Select the **Power Supply** tab. |

Budget:



# Budget on the number of application-specific channels

## At a Glance

A budget on the number of application-specific channels used is made for:

- the processor module of the station
- each module (processor, adapter, I/O module) of the station

This budget is presented in the form of a bar chart where each color has a particular significance, it indicates for each application:

- the number of application-specific channels configured: **green**,
- the number of application-specific channels still available: **white**,
- the excess application-specific channels (not managed by the processor): **red**, when an excess occurs, a message is displayed.

## Processor and adapter budget

This budget shows the number of channels configured by applications **in the station**.

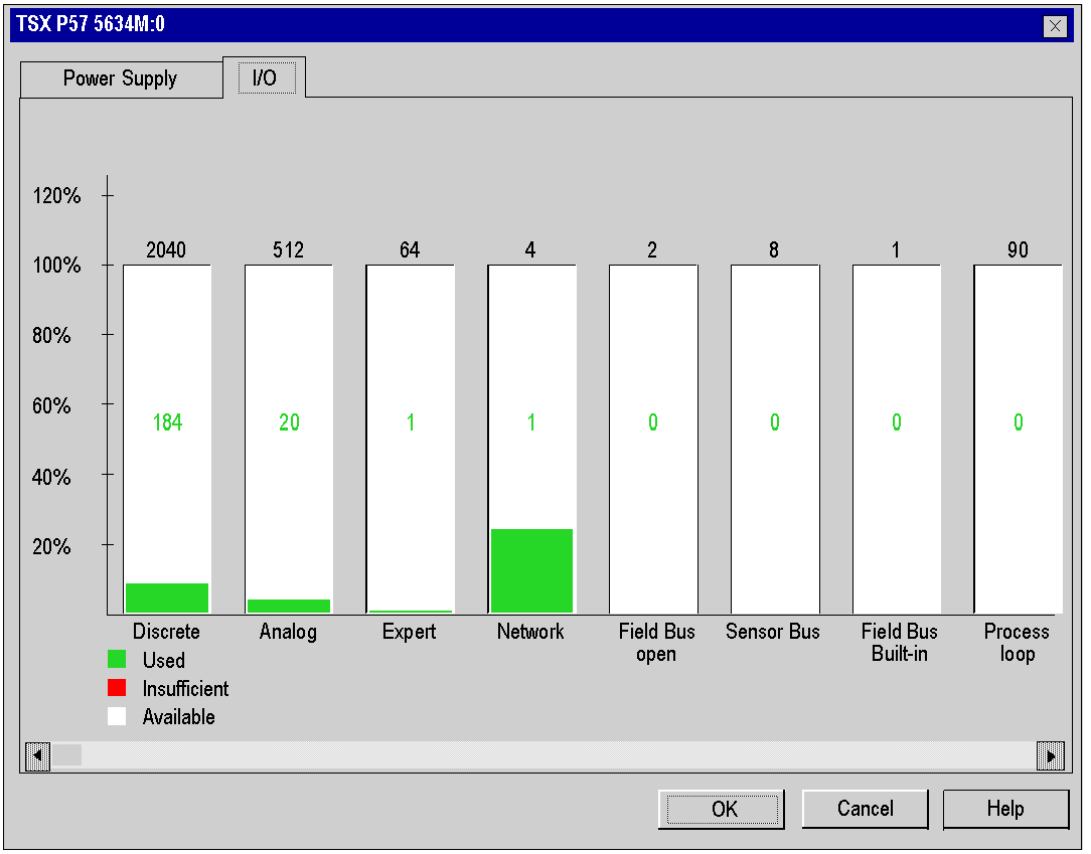
When you add or remove a module, the budget is adjusted on opening the **Power Supply and IO Budget** window.

**NOTE:** The **Power Supply and IO Budget** window must be closed to be able to delete or add a module.

How to access the budget:

| Steps | Action |
|---|---|
| 1 | With the mouse, select the required processor or adapter. |
| 2 | Using the contextual menu, click **Power Supply and IO Budget**. The bar chart is displayed. |
| 3 | Select the **I/O** tab. |

Example of IO budget content:



## Input/output modules budget

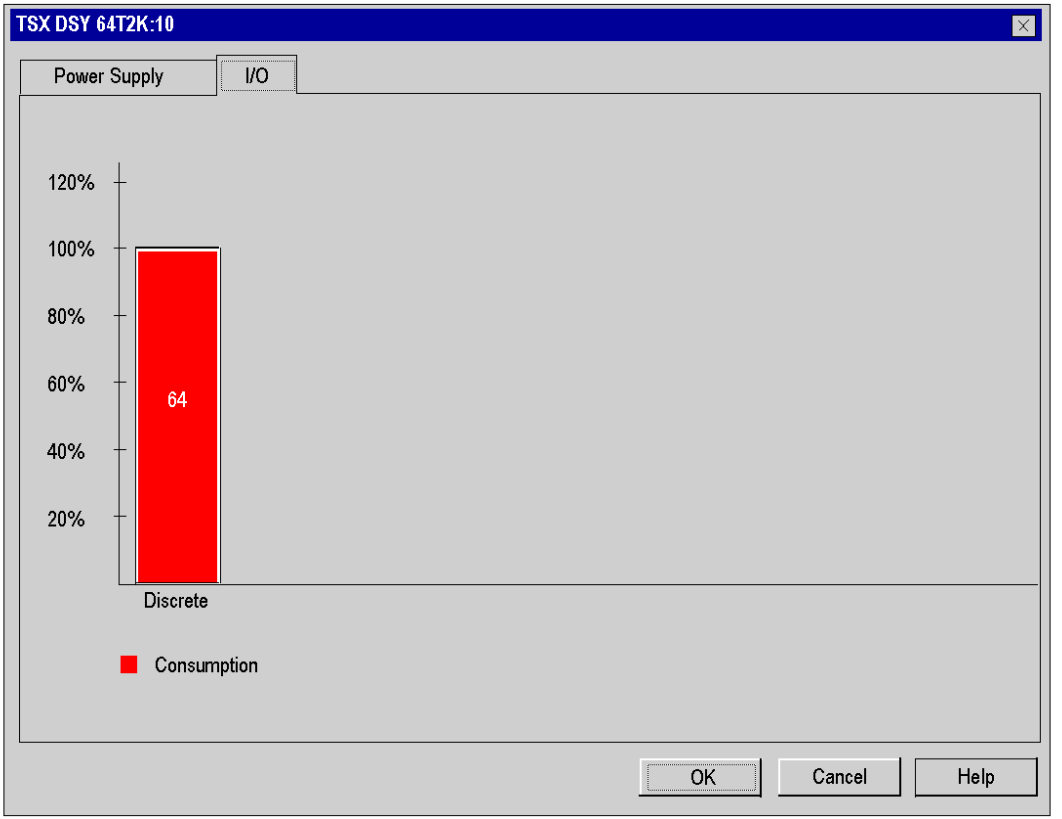
This budget shows the number of application-specific channels configured in the module.

**NOTE:** The **Power Supply and IO Budget** window must be closed to be able to delete or add a module.

How to access the budget of the number of application-specific channels:

| Steps | Action |
|---|---|
| 1 | Using the mouse select the input/output module of your choice. |
| 2 | Using the contextual menu, click **Power Supply and IO Budget**. The bar chart is displayed. |
| 3 | Select the **I/O** tab. |

Budget:



# Section 35.7
## Bus editors in online mode

## Online Information

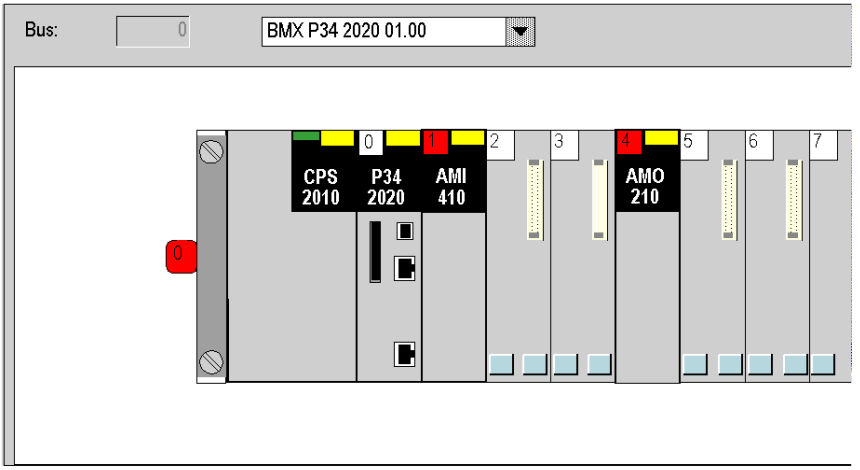
### Animation

The bus editor uses color animation to inform you about the status of a module on the bus.

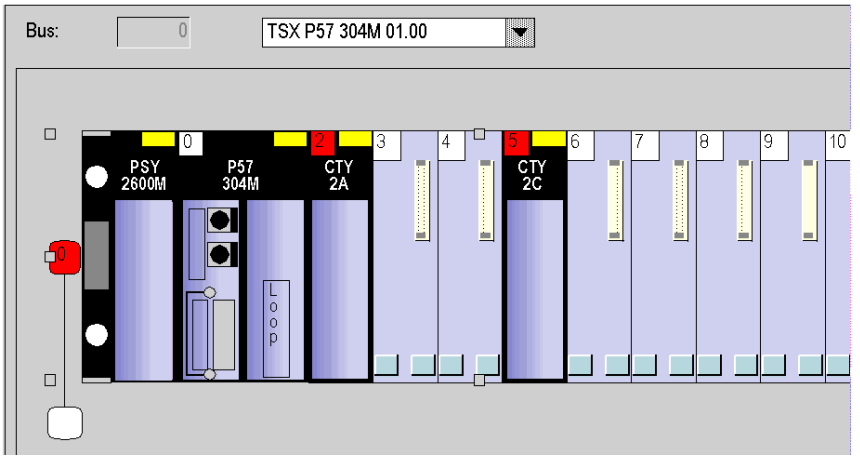
The module's slot number in the rack appears in red if the module is:

- missing,
- faulty,
- badly configured.

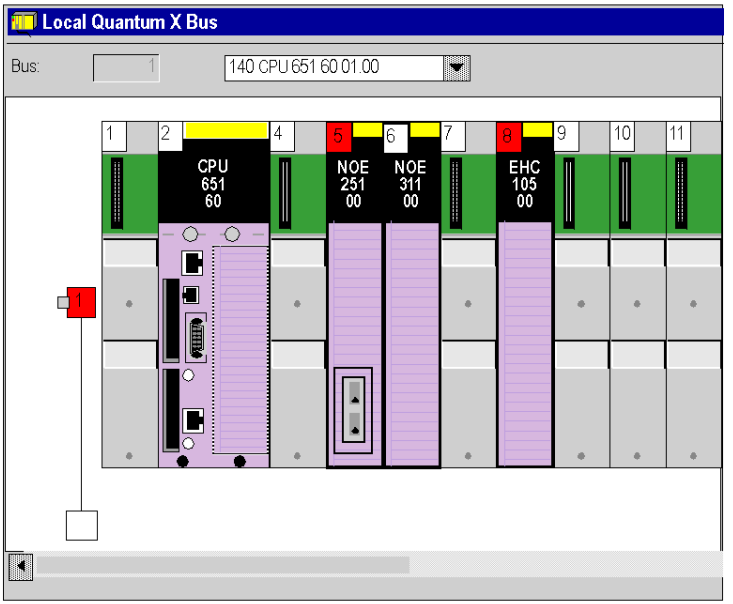
Modicon M340 PLCs



Premium PLCs:




Quantum PLCs:



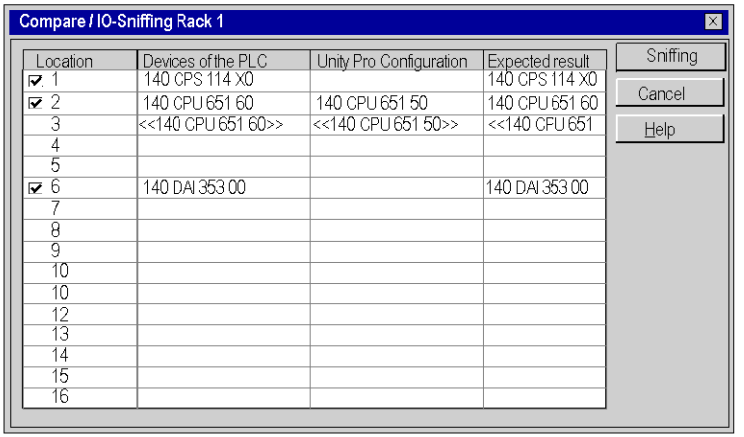

If a fault has occurred on a rack module, the rack number appears in red.

## Automatic detection of the configuration

When connected to a **Quantum** station, the programming software is able to detect the different modules that make up the configuration. This operation can be performed before programming the application, or after the application has been programmed.

## Procedure

The following table describes the procedure for detecting the different elements of a PLC configuration.

| Step | Action |
|---|---|
| 1 | In online mode (PLC in Stop), open the configuration editor. |
| 2 | Select the rack. |
| 3 | From the **Services** menu, select **Compare/IO-Sniffing...**.<br>**Result**: The following window opens.<br>  |
| 4 | Check the **Slot** box if you want:<br>● to add the module detected on initial configuration,<br>● to replace the configured module by the one detected.<br>If the detected module is identical to the configured module, you can keep this module by checking the **Slot** box.<br>If you leave the **Slot** box unchecked, the module specified on initial configuration remains. |
| 5 | Validate the new configuration by clicking **Sniff**. |
| 6 | Click **OK** to confirm your choice.<br>**Result**: The new configuration appears in the configuration editor. |

# Chapter 36
# Modules editor

## Aim of this section

This section guides the user in the configuration of the modules comprising a Premium/Atrium, Quantum or Modicon M340 station.

For Modicon M580 PLC stations, refer to chapter *Configuring the CPU in Unity Pro (see Modicon M580, Hardware, Reference Manual)*.

## What Is in This Chapter?

This chapter contains the following sections:

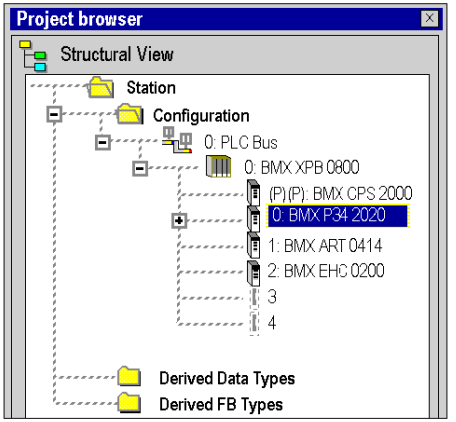

| Section | Topic | Page |
|---|---|---|
| 36.1 | Configuration of Modicon M340 processors | 1352 |
| 36.2 | Configuration of Premium processors | 1363 |
| 36.3 | Configuration of the Quantum processors | 1373 |
| 36.4 | Configuration of the input\output modules for a Premium\Atrium, Quantum or Modicon M340 station | 1383 |

# Section 36.1
# Configuration of Modicon M340 processors

## Configuration of Modicon M340 Processors

### Access to the Configuration Screen

Proceed with the following steps:

| Step | Action |
|---|---|
| 1 | Select the processor.<br>Example:<br>  |
| 2 | Using the contextual menu, click **Open**. |
| 3 | Select the **Configuration** tab. |

## Memory Management

The configuration screen differs, depending on the type of memory management to be used.

The option you select here must comply with the type of memory management you select for the input and output modules *(see page 1220)*.

The following settings are possible:

| If... | Then... |
|---|---|
| **Topological** memory management is configured for the processor | you can only select **Topological** memory management for the discrete and analog modules. |
| **Mixed topological and State RAM** memory management is configured for the processor | you can select either **Topological** or **Mixed topological and State RAM** memory management for the discrete and analog modules. |

The memory management settings of the processor are compared with the memory management settings of the modules during **Analyze**. If any inconsistencies are detected, a appropriate message is displayed in the output window.

The type of memory management you selected for the processor is used as default setting for the **Memory Management** parameter in the **Memory** tab for input / output configuration *(see page 1219)*.

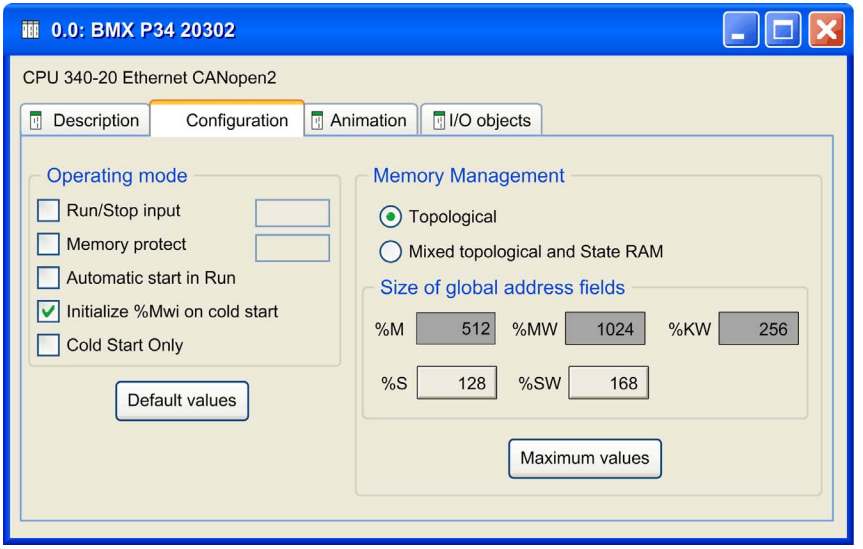
**NOTE:** If you want to import a legacy LL984 Compact application which uses Modbus request to communicate with an HMI, you have to use State RAM addressing to preserve the Modbus exchange between PLC and HMI.

**NOTE:** If you use the fallback configuration in your hardware configuration, take care that the fallback values can only be applied if the **Topological** option is checked.

## Configuration of the Processor

**NOTE:** You can modify the parameters of this dialog box only in offline mode.

Configuration screen for topological memory management:




| ⚠ CAUTION |
|---|
| **LOSS OF DATA ON APPLICATION TRANSFER**<br>Do not press the RESET button on the power supply. Otherwise, `%MWi` is reset and initial values are loaded.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

Proceed with the following steps:

| Step | Action |
|---|---|
| 1 | If you wish, enable the **Run/Stop input** option *(see page 1358)*.<br>Do not enable this option if the associated discrete input is mapped in State RAM because this inhibits the PLC start-up. |
| 2 | If you wish, enable the **Memory protect** option.<br>The protection is activated by an input bit. The protection level depends on the Firmware versions, see detailed information *(see page 1358)*. |
| 3 | If you wish, enable the **Automatic start in Run** option *(see page 1359)*. |

| Step | Action |
|---|---|
| 4 | Confirm whether the internal words %MWi *(see page 1359)* are to be initialized on cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )* triggered by software (application download, initialize command, restore command, %S0 activation, cold start button on the PLC screen) or not. By checking the box, the PLC writes the initial values to %MWI. |
| 5 | If you wish, enable the **Cold Start Only** option *(see page 1359)*. |
| 6 | Define the type of **Memory Management**:<br>● select the option **Topological** for topological addressing<br>● select the option **Mixed topological and State RAM** for a mixture of topological and defined State RAM addressing |
| 7 | Proceed with address configuration for the 2 different addressing types as described in the following separate descriptions. |

**NOTE:** The State RAM address starts at address 0, but the first available address for discrete and analog modules is %I1, %M1, %IW1, and %MW1.

## Topological Addressing

Proceed with address configuration for topological addressing as follows:

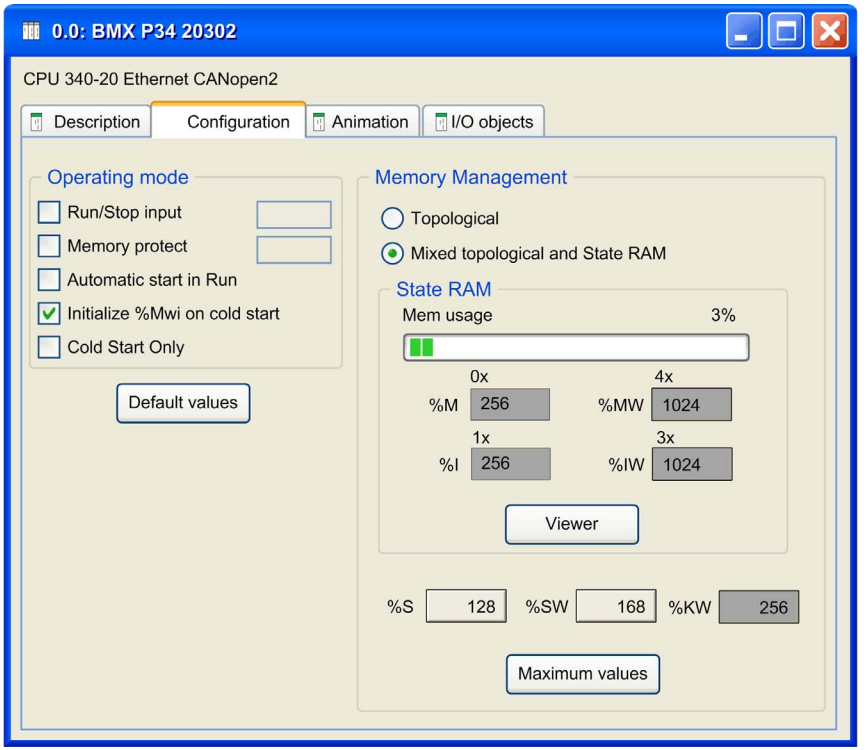
| Step | Action |
|---|---|
| 1 | In the **Size of global address fields** of the dialog box define for the application *(see Unity Pro, Program Languages and Structure, Reference Manual )* the number of internal bits %M. |
| 2 | Define the number of internal words %MW. |
| 3 | Define the number of constants %KW. |

The system bits and system words are fixed by the manufacturer.

Unity Pro Modicon M340 applications of version 6.0 and earlier are automatically configured with this address configuration for topological addressing.

## Mixed Topological and State RAM Addressing

Configuration screen for mixed topological and State RAM memory management:




For mixed topological and State RAM addressing, the following options are additionally available in the configuration screen:

| Option | Value | Description |
|---|---|---|
| **Mem usage** | The value (expressed as a percentage and displayed on the scale bar) depends on the memory usage of the Modicon M340 Hot Standby configuration. | A bar graph displays the percentage of the memory already in use. |
| **%M-0x** | Enter the appropriate values. The values depend on the Hot Standby configuration. | Size of the different memory areas.<br>**NOTE:** The values for **%IW** and **%MW** have to be divisible by 8. |
| **%MW-4x** | | |
| **%I-1x** | | |
| **%IW-3x** | | |
| **Viewer** | – | Opens the State RAM Viewer tab *(see page 1360)* which displays the allocation of used memory. |

**NOTE:** The State RAM address starts at address 0, but the first available address for discrete and analog modules is `%I1`, `%M1`, `%IW1`, and `%MW1`.

**NOTE:** State RAM addresses are not checked during input. If any range overlapping is detected or any address exceeds the range of the State RAM, an appropriate message is displayed at the validation screen or during **Analyze**.

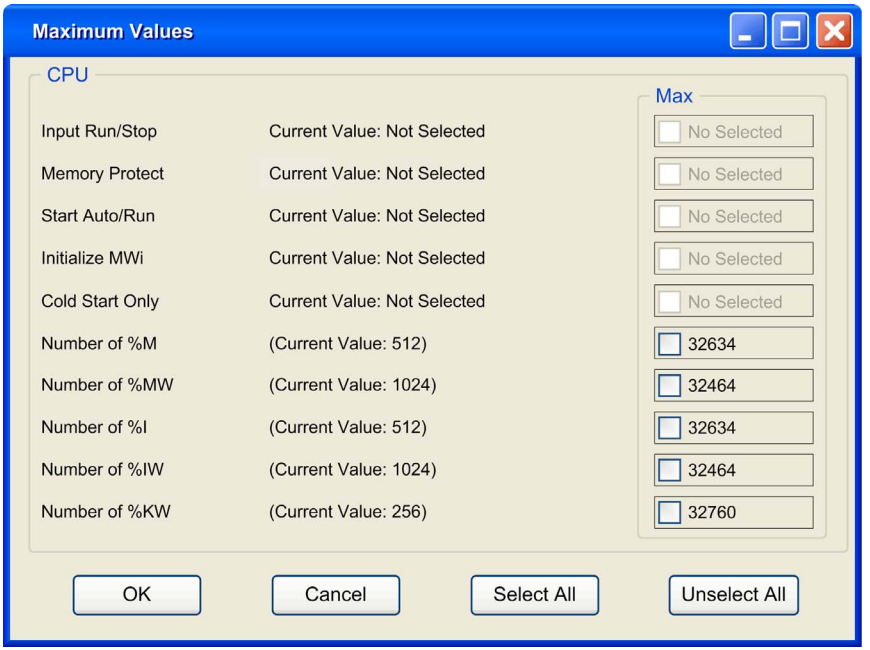
**NOTE:** With this type of memory management, all channels are associated to one task (MAST or FAST).

## Pre-set values

To select the:

- **default** values, press the **Default values** button,
- **maximum** values, press the **Maximum values** button,

Window displayed by pressing the **Default values** button:



Use of the command buttons:

| Button | Role |
|---|---|
| Unselect All | Is used to invalidate all the default values posted on the right of the check boxes if they have been selected. |
| Select All | Is used to select all the default values posted on the right of the check boxes. |
| Cancel | Is used to exit. |
| OK | Is used to exit taking the values into account. |

**NOTE:** With the **Maximum values** button, the window displayed is the same, only the values on the right of the check boxes are different.

### Input RUN/STOP

The input **%Ir.m.c** can be parameterized to switch the PLC to **RUN/STOP** mode in the following way:

- `%Ir.m.c` to 1 -> the PLC switches to RUN (execution of the program),
- `%Ir.m.c` to 0 -> the PLC switches to STOP mode (stop program execution).

**NOTE:** A STOP command always takes priority over a RUN command. A STOP command sent from a terminal or via the network has priority over the `%Ir.m.c` input.
An error on the RUN/STOP input causes a switch to STOP.
Do not enable this option if the associated discrete input is mapped in State RAM because this inhibits the start-up of the PLC.

### Memory protect

The input **%Ir.m.c** can be parameterized to protect the internal application ram and the memory card in the following way:

- `%Ir.m.c` to 0 -> the internal application and the memory card are not protected,
- `%Ir.m.c` to 1 -> the internal application and the memory card are protected.

For Firmware versions < 2.60, the Memory Protect function prohibits the transfer of a project into the PLC and modifications in online mode, but RUN/STOP commands are available.

For Firmware versions greater and equal to 2.60, the Memory Protect function prohibits the transfer of a project into the PLC and modifications in online mode, **for cybersecurity improvement, the connection to the PLC is forbidden then the RUN/STOP command are not available**.

**NOTE:** If the input is in error (Input Module not physically plugged, wrong wiring, input not powered...), %Ir.m.c is considered at 1 (memory is protected). To remove this protection in this configuration screen, the input should not be in error.

### Automatic start in Run

The enabling of this option automatically changes the PLC to RUN mode at the time of a cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

### Initialize %MWi

On a cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )* or on download if you check the box (default state):

- the %MWi are handled like other global variables (initialized to 0 or initial value, according to current application) in all cold start cases,

On cold start or on download if you uncheck the box:

- if %MW were previously saved in internal flash memory (using the %SW96 word) they are restored from internal flash memory,
- if not,
  - if cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )* is linked to a power-off or of a push on the reset button, the %MW are initialized:
  - if not, the current values of %MW are maintained.

**NOTE:** if the new (or restored) application has more %MW than the previous one, the added %MW are set to 0 (non-zero initial values are not applied).

### Cold Start Only

If checked, this option forces the cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )* of the application, instead of the normal warm start *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

By default, the Cold Start Only option is unchecked.

The Cold Start Only option is only supported since PLC V2.1.

An application using this functionality is not:

- downloadable on a PLC with a previous version,
- executable on a PLC with a previous version,
- usable with Unity Pro V4.0 or lower.

**NOTE:** The Cold Start Only check box is present only if the current selected PLC can support it.

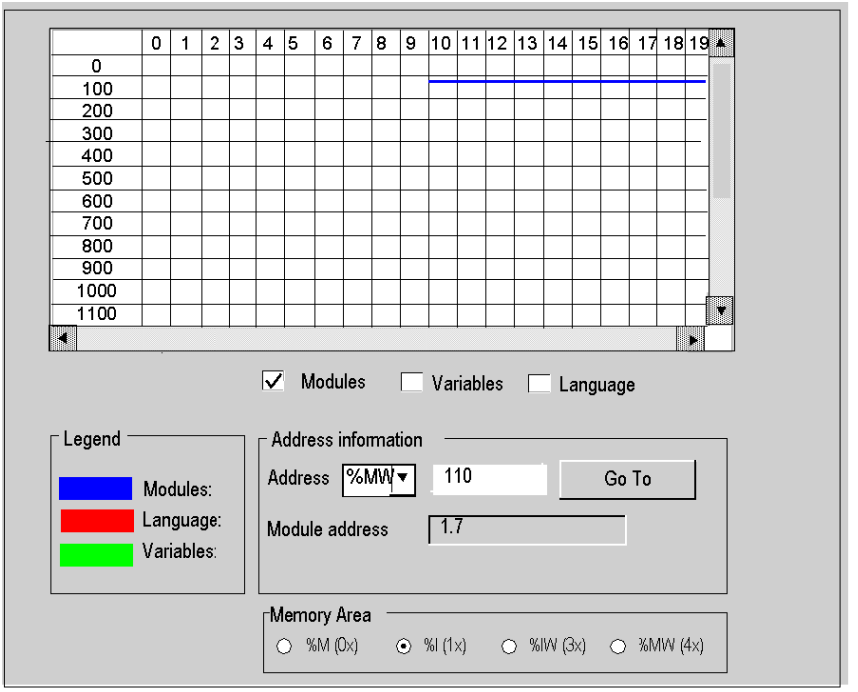
### Memory Management

This area of the dialog box allows you to define the type of addressing (topological or mixed addressing).

For mixed addressing it contains a bar graph indicating the size of the **State RAM** memory used in your project in relation to the maximum memory size and a **Viewer** button opening the State RAM memory viewer.

## State RAM Memory Viewer

The following screen displays the content of the State RAM memory areas of the M340 processor:




**NOTE:**
The State RAM viewer is directly accessible via the menu:

- **PLC → State Ram Viewer**

Description of the screen zones

| Zone | Description |
|---|---|
| Table | This table represents the mapping of the memory area selected with the **Memory Area** selection buttons.<br>The occupied memory addresses are marked by dashes:<br>● blue: for module type objects<br>● red: for objects entered in the program<br>● green: for objects defined in the variable editor<br>The scroll bars can be used to access the entire memory area. |
| **Modules**, **Variables** and **Language** checkboxes | These checkboxes are used to filter the information displayed in the table. If, for example, **Modules** is the only checkbox that is selected, only module-type information is displayed. |

| Zone | Description |
|---|---|
| **Address information** | By entering the object and its address in the **Address** fields and clicking on the **Go To** button, you display directly the address in the table (without using the scroll bars). This zone also has a display function; it displays the object and address of the box selected in the table. For module-type objects, it also gives the topological address of the module concerned. |
| **Memory Area** selection buttons | These buttons are used to select the memory area to display in the table:<br>● **%M (0x)** output bits and internal bits<br>● **%I (1x)** input bits<br>● **%IW (3x)** input words<br>● **%MW (4x)** output words and internal words |

## Size of Located Data in Case of State RAM

The following table shows maximum and default size of located data in case of State RAM configuration according to the type of processor.

| Type of Objects | Address | BMX P34 1000 V2.40Processor | | BMX P34 2000, 20102, 2020, 20302 Processors (all V2.40) | |
|---|---|---|---|---|---|
| | | Maximum Size | Default Size | Maximum Size | Default Size |
| output bits and internal bits | %M (0x) | 32765 | 752 | 65530 | 1504 |
| input bits and internal bits | %I (1x) | 32765 | 752 | 65530 | 1504 |
| input words and internal words | %IW (3x) | 32765 | 256 | 65530 | 512 |
| output words and internal words | %MW (4x) | 32765 | 256 | 65530 | 512 |

**NOTE:** To use State RAM configuration you need Unity Pro 6.1 or later and Modicon M340 firmware 2.4 or later.

## Impact of a Cold Start on %I and %IW Areas

After a Cold Start or %S0 setting, the %I and %IW areas are set to 0 and forcing state is reset.

### CPU Modbus Server Uses %I and %IW Areas

From Unity Pro V6.1:

- When the State RAM **is** configured in a CPU, the Modbus server access to %I area for all input bits operations and to %IW area for all input words operations.
  Base address %SW138 and %SW140 are not used in this case.
  The first object address in a Modbus request (0000) corresponds to %I1 or %IW1.
- With Modicon M340 using firmware V2.4 or later, when the State RAM **is not** configured, the Modbus server behaves as in firmware V2.3.

Base address for %M and %MW in system words %SW139 and %SW141 are used whether State RAM is configured or not.

# Section 36.2
## Configuration of Premium processors

### Aim of this sub-section

This sub-section guides the user in the configuration of processor modules for Premium stations.

### What Is in This Section?

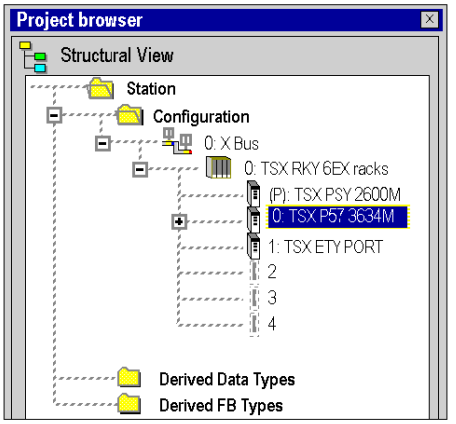
This section contains the following topics:


| Topic | Page |
|---|---|
| Configuration of Premium\Atrium processors | 1364 |
| Access to PCMCIA memory cards of Premium processors | 1370 |
| Access to control loop configuration for Premium processors | 1372 |

# Configuration of Premium\Atrium processors

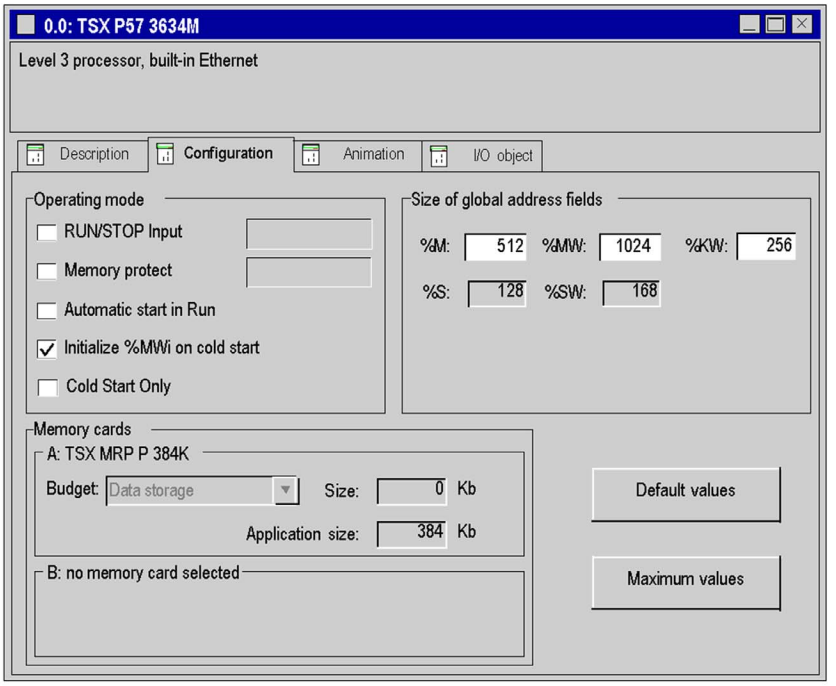
## Access to the configuration screen

Proceed with the following steps:

| Step | Action |
|---|---|
| 1 | Select the processor.<br>Example:<br>  |
| 2 | Using the contextual menu, click **Open**. |
| 3 | Select the **Configuration** tab. |

## Configuration of the processor

Configuration Screen:



Proceed with the following steps:

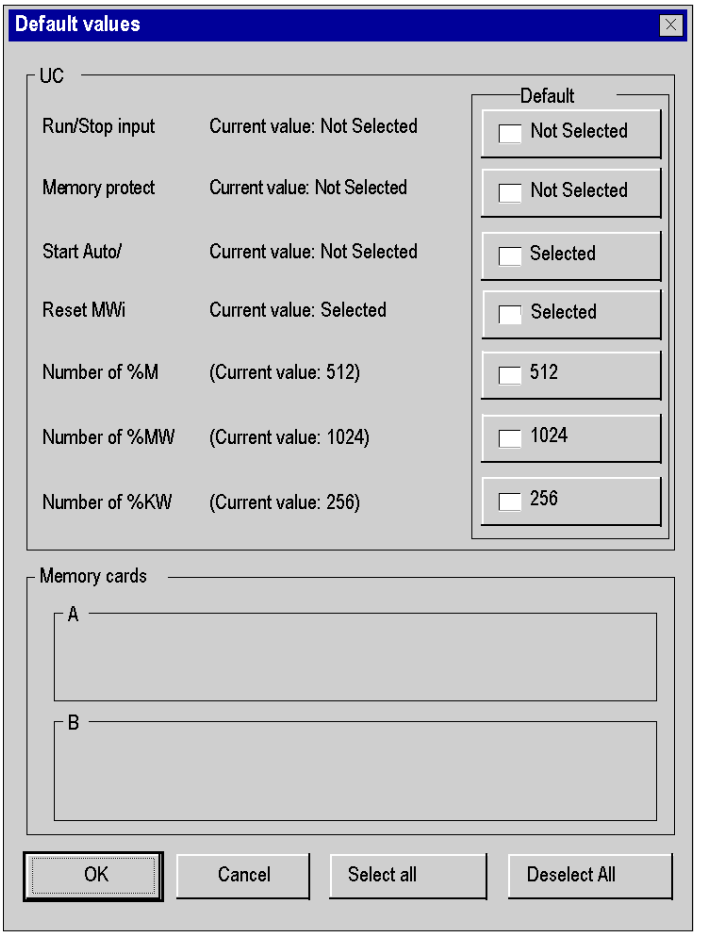
| Step | Action |
|---|---|
| 1 | If you wish, enable the RUN/STOP input *(see page 1368)*. |
| 2 | If you wish, enable the Memory Protect.<br>The protection is activated by an input bit. It prohibits the transfer of a project into the PLC and modifications in online mode, regardless of the communication channel. The Run and Stop commands are authorized. |
| 3 | If you wish, enable the Automatic start in Run *(see page 1368)*. |
| 4 | Confirm whether the internal words %MWi *(see page 1369)* are to be initialized (on cold restart, on download, after the insert PCMCIA memory card) or not. |
| 5 | If you wish, enable the Cold Start Only *(see page 1369)* feature. |
| 6 | Define for the application: *(see Unity Pro, Program Languages and Structure, Reference Manual )*<br>● the number of internal bits %M,<br>● the number of internal words %MW,<br>● the number of constants %KW,<br>The bits and system words are fixed by the manufacturer. |
| 7 | The **Memory cards** field is completed according to the PCMCIA memory card selected *(see page 1370)*. The data storage area is only used by EFBs such as `READ_U_PCMCIA` and `WRITE_U_PCMCIA`. The application area is used for the program. However, some parts of the program (located and unlocated variables, FB instances) are always present in the CPU internal memory *(see page 147)*. |

## Pre-set values

To select the:

- **default** values, press the **Default values** button,
- **maximum** values, press the **Maximum values** button,

Window displayed by pressing the **Default values** button:



Use of the command buttons:

| Button | Role |
|---|---|
| Deselect All | Is used to invalidate all the default values posted on the right of the check boxes if they have been selected. |
| Select All | Is used to select all the default values posted on the right of the check boxes. |
| Cancel | Is used to exit. |
| OK | Is used to exit taking the values into account. |

**NOTE:** With the **Maximum values** button, the window displayed is the same, only the values on the right of the check boxes are different.

### RUN/STOP input

The input **%Ir.m.c** can be parameterized to switch the PLC to **RUN/STOP** mode in the following way:

- %Ir.m.c to 1 -> the PLC switches to RUN (execution of the program),
- %Ir.m.cl to 0 -> the PLC switches to STOP mode (stop program execution).

**NOTE:** A STOP command by the input %Ir.m.c is priority compared to a RUN operation by terminal or by network command.
An error on the RUN/STOP input causes a switch to STOP.

### Memory protect

The input **%Ir.m.c** can be parameterized to protect the internal application ram and the memory card in the following way:

- %Ir.m.c to 0 -> the internal application and the memory card are not protected,
- %Ir.m.c to 1 -> the internal application and the memory card are protected.

**NOTE:** If the input is in error, %Ir.m.c is considered at 1 (memory is protected). To remove this protection in this configuration screen, the input should not be in error.

### Automatic start in Run

The enabling of this option automatically changes the PLC to RUN mode at the time of a cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

Two types of start:

- in the absence of a PCMCIA memory card, the PLC starts on the contents of the internal RAM of the processor,
- in the presence of a PCMCIA memory card it is its content which fixes the start.

### Initialize %MWi

On cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )*, on download or after the insert of PCMCIA memory card:

Depending of the flag:

| | | **At the end of the transfer : ==> The initial values are assigned to the variables.** | **On setting of %S0 : ==> The initial values are assigned to the variables.** |
|---|---|---|---|
| Not checked | OS >= V2.3 | **NO** | **NO** |
| | OS < V2.3 | **YES** | **NO** |
| Checked | OS >= V2.3 | **YES** | **YES** |
| | OS < V2.3 | **YES** | **YES** |

the %MWi values will be initialized to 0 or initial value even if the box is unchecked,

- if you change the %MW area on the field **Size of global address fields**,
- if after a power restoration, the OS detects that the contents of the internal RAM has not been saved (PLC powered off and Back-up battery non-operational).

### Cold Start Only

If checked, this option forces the cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )* of the application, instead of the normal warm start *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

By default, the Cold Start Only option is unchecked.

The Cold Start Only option is only supported since PLC V2.7.

An application using this functionality will not be:

- downloadable on a PLC with a previous version,
- executable on a PLC with a previous version,
- usable with Unity Pro V4.0 or lower.

**NOTE:** The Cold Start Only check box is present only if the current selected PLC can support it.

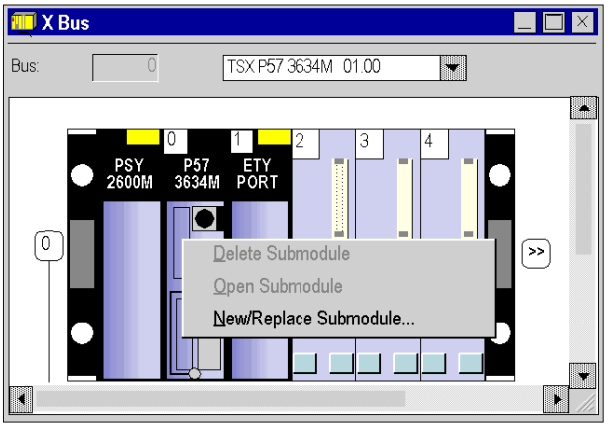
# Access to PCMCIA memory cards of Premium processors

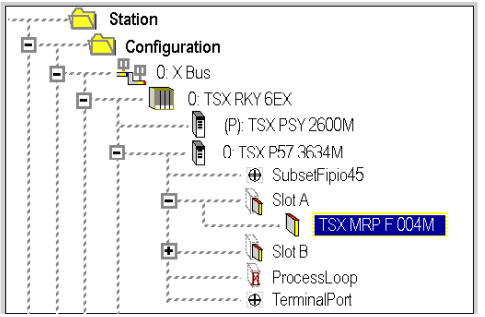
## Introduction

This operation describes how to choose a PCMCIA memory card.

## Selection of a PCMCIA card

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Access the rack configuration screen. |
| 2 | Select the PCMCIA location and from the popup menu execute the **New/Replace Submodule...** command.<br>Example:<br>  |
| 3 | Select the PCMCIA family, and the card within the family. |

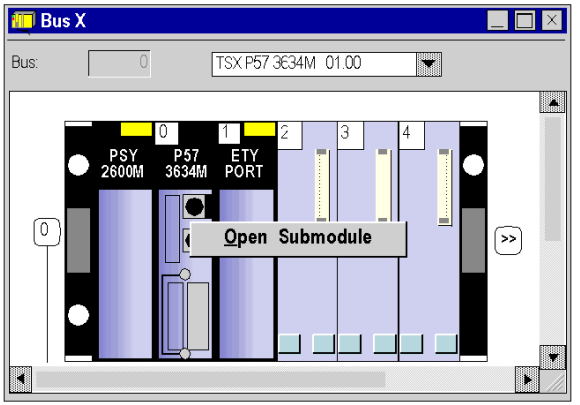
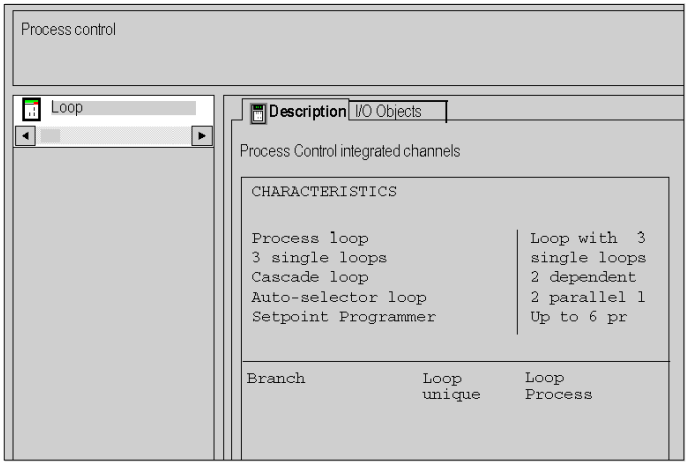
| Step | Action |
|---|---|
| 4 | Confirm with **OK**<br>The card is added to the directory tree of the processor:<br>  |

To delete a PCMCIA memory card, select its location and from the popup menu execute the **Delete Submodule** command.

# Access to control loop configuration for Premium processors

## Instructions

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Access the rack configuration screen. |
| 2 | Select the location of the process control loops and from the popup menu execute the **Open Subobject** command.<br>Example:<br>  <br><br>The following screen is displayed:<br>  <br> |
| 3 | Configure the process control loops *(see Premium and Atrium using Unity Pro, Process Control, User Manual)*. |

# Section 36.3
## Configuration of the Quantum processors

### Aim of this sub-section

This sub-section guides the user in the configuration of the processor modules for the Quantum stations.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Configuration of Quantum processors | 1374 |
| Configuration of Modbus ports on Quantum | 1380 |

# Configuration of Quantum processors

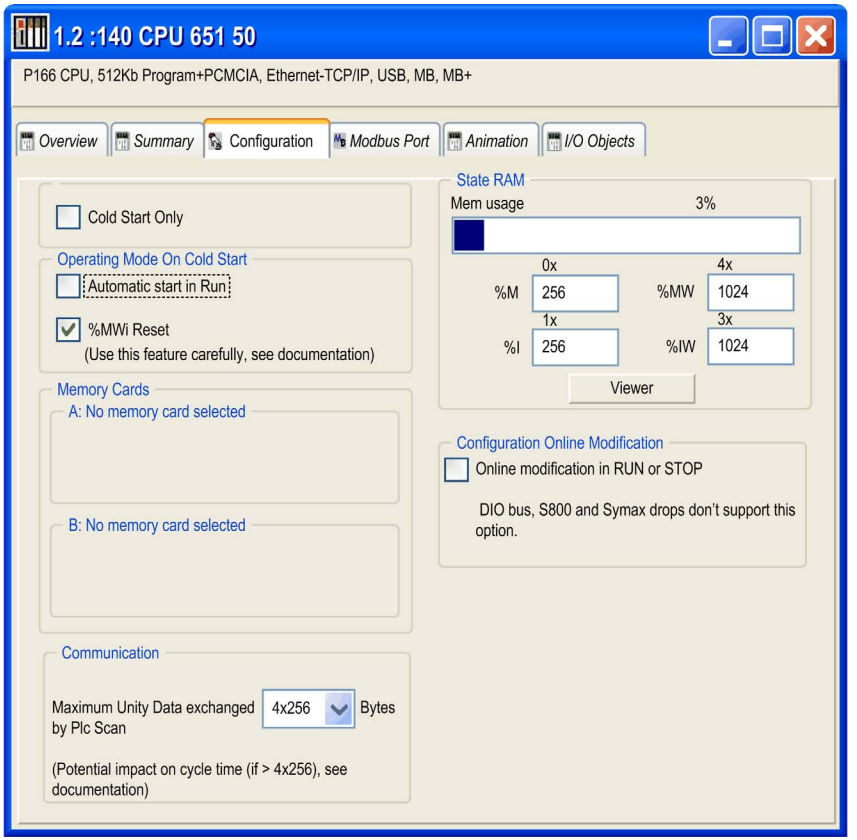
## Accessing the configuration screen

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Access the rack configuration screen. |
| 2 | Select the processor. |
| 3 | Using the contextual menu, click **Open Module**. |
| 4 | Select the **Configuration** tab. |

## Configuring the processor

The following figure shows an example of the Quantum processor configuration screen:



Description of the Configuration Screen:

| Item | Option | Value | Description |
|---|---|---|---|
| Operating Mode On Cold Start | Automatic start in Run | x | Determines the operating condition during Cold Start |
| | %MWi Reset on cold start | x | |
| | Cold Start Only | x | If you wish, enable the Cold Start Only *(see page 1377)* feature. |
| Memory Cards | A: | N/A | Displays the configuration in the PCMCIA Slots |
| | B: | N/A | |
| Communication | By default, the bandwidth is 4x256 bytes, supported by the OS versions prior to V2.80 for the CPU and V4.60 for the NOE. | | The maximum data volume exchanged each cycle between the NOE and CPU modules. |
| | For Quantum processors:<br>• 140 CPU 311 10<br>• 140 CPU 534 14<br>• 140 CPU 434 12 | 4x256<br>4x1024 | |
| | For Quantum processors:<br>• 140 CPU 651 50<br>• 140 CPU 651 60<br>• 140 CPU 652 60<br>• 140 CPU 670 60<br>• 140 CPU 671 60<br>• 140 CPU 672 60<br>• 140 CPU 672 61<br>• 140 CPU 678 61 | 4x256<br>4x1024<br>8x1024<br>12x1024 | |
| State RAM | Mem usage | 1. | A bar displays percent of memory used. |
| | %M-0x | 2. | Size of the different memory areas<br>**Note:** The values for %IW and %MW have to be divisible by 8. |
| | %MW-4x | 2. | |
| | %I-1x | 2. | |
| | %IW-3x | 2. | |
| | Viewer | N/A | Opens the State RAM Viewer tab, which displays the allocation of used memory. (See the illustration following.) |

| Item | Option | Value | Description |
|---|---|---|---|
| Configuration Online Modification | Online modif in RUN or STOP | x | This check box allows you to:<br>● Add or delete discrete or analog modules,<br>● Modify Parameters<br>**NOTE:** These modifications can be done in RUN. |
| 1. The value (expressed as a percentage and displayed on the scale) depends on the memory usage of the Hot Standby configuration.<br>2. Enter the appropriate values. All values depend on Hot Standby configuration. | | | |

### Automatic start in RUN

The enabling of this option automatically changes the PLC to Run mode *(see Unity Pro, Program Languages and Structure, Reference Manual )*on cold start.

Two types of start:
- in the absence of a PCMCIA memory card, the PLC starts on the contents of the internal RAM of the processor,
- in the presence of a PCMCIA memory card it is its content which fixes the start.

**⚠ WARNING**

**UNWANTED APPLICATION RUN ON PLC COLD START**

With the **Automatic start in RUN** option enabled, the following events will trigger the run of the application on cold start:
- Inserting the PCMCIA card when the PLC is powered
- Replacing the processor while powered
- Unintentional or careless use of the reset button
- Powering up a PLC with a defective battery after a power outage

To prevent the run of the application on cold start:
- use the STOP input (on Premium PLCs)
- use the switch on the front panel of the processor (for Quantum PLCs)

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

### %MWi Reset

On application download:
- if you check the box, the %MWi values will be re initialized or set to 0,
- if you uncheck the box, the %MWi values will set to 0.

On cold start or after inserting the PCMCIA memory card:
- if you check the box, the %MWi values will be re initialized or set to 0,
- if you uncheck the box, the %MWi values will retain their current value.

### Cold Start Only

If checked, this option forces the cold start *(see Unity Pro, Program Languages and Structure, Reference Manual )* of the application, instead of the normal warm start *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

By default, the Cold Start Only option is unchecked.

The Cold Start Only option is only supported on High End PLC since V2.7.

An application using this functionality will not be:
- downloadable on a PLC with a previous version,
- executable on a PLC with a previous version,
- usable with Unity Pro V4.0 or lower.

**NOTE:** The Cold Start Only check box is present only if the current selected PLC can support it.

### Communication

When the **UNITY** protocol under TCP/IP is used (OFS or Unity Pro), it is possible to configure the maximum volume of data that can be exchanged each cycle between the CPU and the NOE modules using the **Maximum** Unity Data exchanged by Plc Scan **option**.

This functionality is only supported on CPU modules with OS version 2.80 or higher, and on NOE modules with OS version 4.60 or higher.

The bandwidth set is valid between the CPU and all existing NOE modules. It is not possible to set different bandwidths for each of the modules.

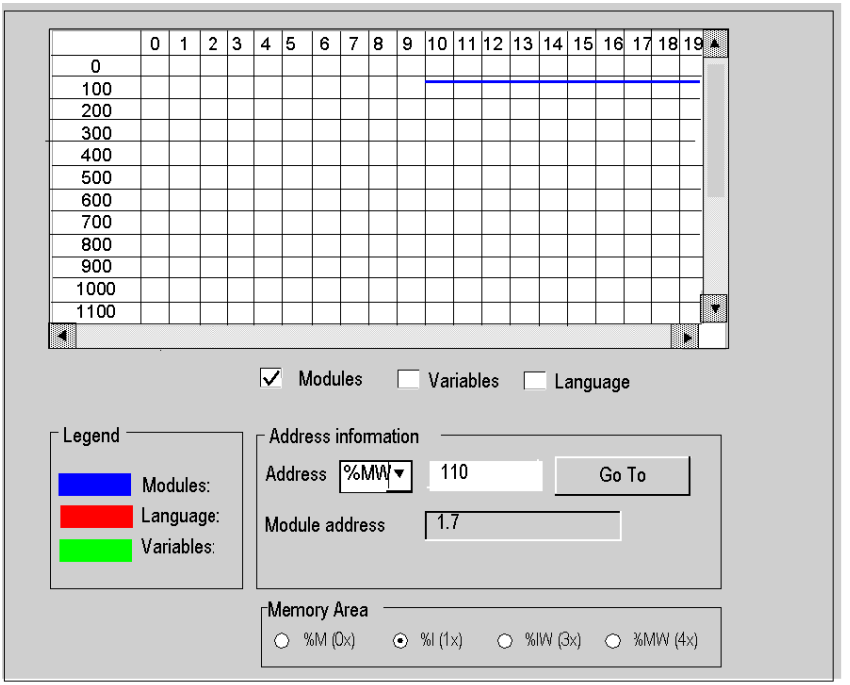
Increasing this bandwidth has an impact on the cycle time of the controller (2ms per kbytes exchanged). This impact is proportional to the amount of data actually exchanged and not the bandwidth configured. So, if the channel is set to the maximum, but not used, the impact on cycle time will be negligible.

### State RAM memory

The **State RAM** bar chart allows you to know the size of the **State RAM** memory used in your project in relation to the maximum memory size.

## State RAM memory Viewer

The following screen displays the content of the State RAM memory areas of the Quantum processor:




**NOTE:**
The state ram viewer can be directly accessible via the menu:

- **PLC → State Ram Viewer**

Description of the screen zones

| Zone | Description |
|---|---|
| Table | This table represents the mapping of the memory area selected in the **Memory Area** field.<br>The occupied memory addresses are marked by dashes:<br>● blue: for module type objects<br>● red: for objects entered in the program<br>● green: for objects defined in the variable editor<br>The scroll bars can be used to access the entire memory area. |
| **Modules**, **Variables** and **Language** checkboxes | These checkboxes are used to filter the information displayed in the table. If, for example, 'module' is the only thing checked, only module-type information will be displayed. |

| Zone | Description |
|---|---|
| **Address information** | By entering the object and its address in the **Address** fields and clicking on the **Go to** button, you display directly the address in the table (without using the scroll bars). This zone also has a display function; it displays the object and address of the box selected in the table. For module-type objects, it also gives the topological address of the module concerned. |
| **Memory Area** selection buttons | These buttons are used to select the memory area to display in the table:<br>● **%M (0x)** output module bits and internal bits<br>● **%I (1x)** internal module bits<br>● **%IW (3x)** input/output register words and internal words<br>● **%MW (4x)** I/O module output words and internal words |

## Configuration online modification

On PLCs supporting this functionality a check box is activated and appears in the CPU Editor tab.

If the **Online modification in RUN or STOP** check box is selected, the user can do online RUN modification like addition / deletion, and modification of the module parameters.

# Configuration of Modbus ports on Quantum

## Assignment of Modbus ports

The Modbus port configuration screen can be used to select the Modbus communication parameters for 3 ports: ports No. 1, 2 and 3.

Port No. 1 is assigned:

- to the built-in Modbus port for 140 CPU 6•• processors
- to the first built-in Modbus port (MODBUS COMM1) for other Quantum processors

Port No. 2 is assigned:

- either to the second built-in port (MODBUS COMM2), for processors with 2 built-in ports
- or to one or several NOM modules.

Port No. 3 is assigned: to one or several NOM modules

**Rule for assignment of ports 2 and 3 to NOM modules:**

Starting with the lowest slot numbers:

- the first NOM module will be assigned to port 2,
- the second will be assigned to port 3,
- the third will be assigned to port 2,
- the fourth will be assigned to port 3,
- the fifth will be assigned to port 2,
- the sixth will be assigned to port 3.

**Note:** It is possible to change the assignment order of 2 of them by entering

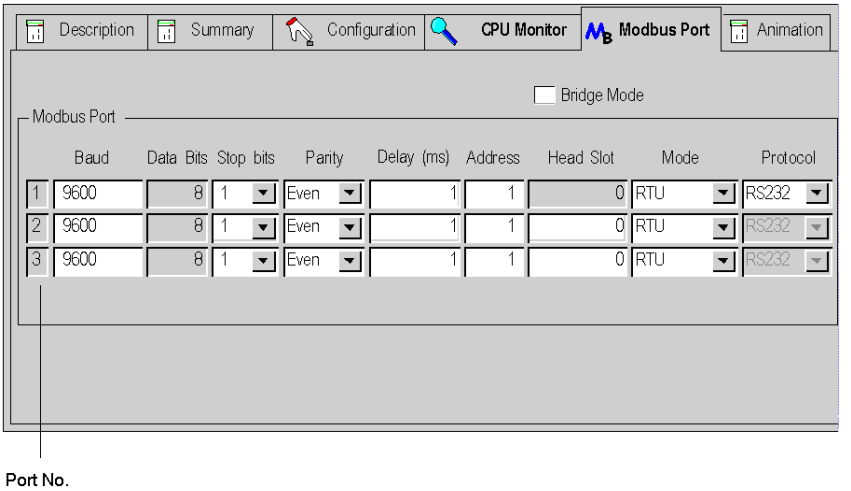
their slot number in the **Head Slot** column (see table below).

**Important:**

The port parameters are only taken into account if the switches on the front panels of Quantum 140 CPU 3••/4••/5•• processors (used with first built-in port MODBUS COMM1) and of NOM modules are in the **Mem** position.

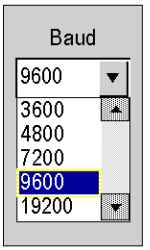
## Access to the configuration editor

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the processor and then select the **Open module** command from the popup menu. |
| 2 | Select the **Modbus Port** tab.<br>**Result**: The following screen appears:<br>  <br> |

## Presentation of configuration parameters

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Choose the operating mode: **Bridge Mode** checkbox |
| 2 | Choose the transmission speed:<br>  <br> |

| Step | Action |
|---|---|
| 3 | Choose the number of Stop bits:<br>  |
| 4 | Choose the parity:<br>  |
| 5 | Choose the Delay which corresponds to the minimum time tolerated between two frames, also called delay between frames. |
| 6 | Choose the Modbus port slave address. |
| 7 | By default, if the **Head Slot** fields are left set to 0, the NOM modules observe the normal assignment rule described above.<br>Entering the slot number of a NOM module assigns it to the corresponding port (the assignment rule is no longer applied for this module but continues to apply to other modules). |
| 8 | Choose the communication mode:<br>  |
| 9 | Choose the communication protocol:<br>  |

# Section 36.4
## Configuration of the input\output modules for a Premium\Atrium, Quantum or Modicon M340 station

### Subject of this Section

This section guides the user through the configuration procedure for input\output modules dedicated to a Premium\Atrium, Quantum or Modicon M340 station.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Access to the Modicon, Premium or Atrium I/O module configuration editor | 1384 |
| Access the Quantum input/output module configuration editor | 1386 |

# Access to the Modicon, Premium or Atrium I/O module configuration editor

## Procedure

Carry out the following actions:

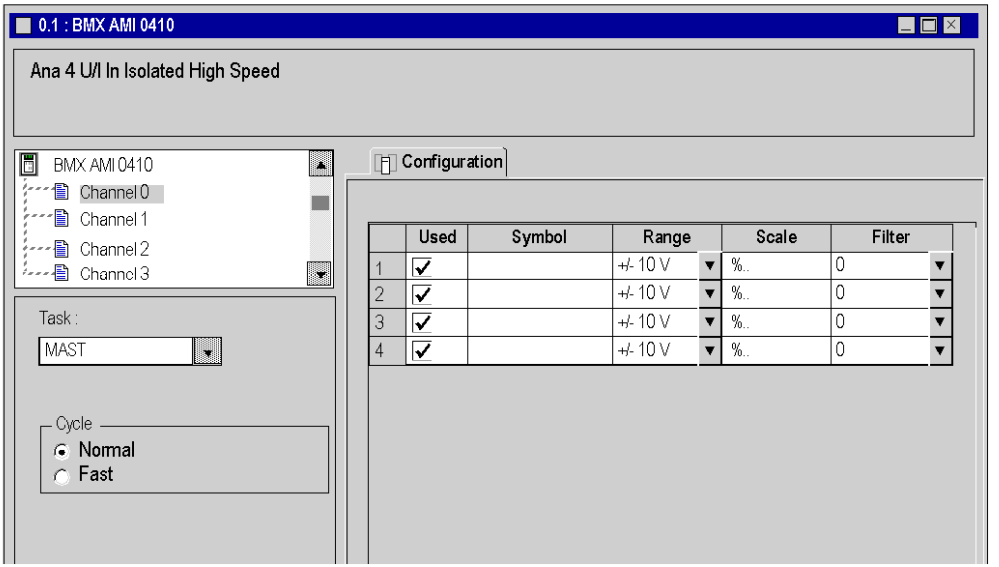
| Step | Action |
|---|---|
| 1 | With the mouse, select the module to be configured. |
| 2 | Using the contextual menu, click **Open Module**. |

## Configuration in offline mode

The configuration window is an editor from which you configure the module, some of your entries are checked, and in some cases you are assisted.

To complete the various fields, **refer to the application-specific manual corresponding to the module** in which you will find the procedure and information necessary for the configuration.

Screen example for a discrete module:



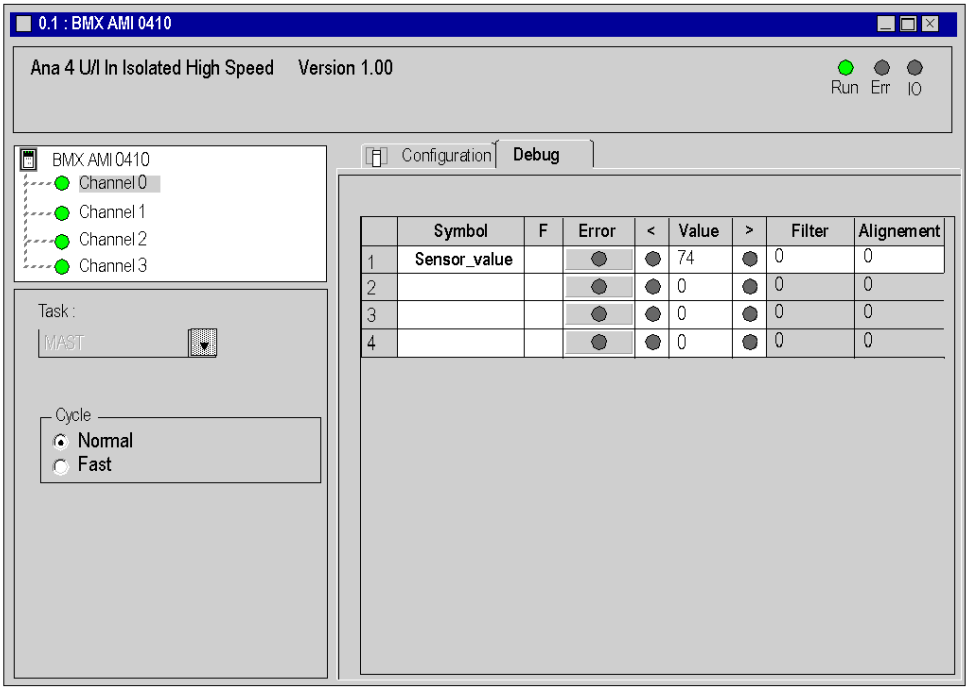
## Configuration in online mode

In online mode two additional tabs are accessible:

- The **Debug** tab, used to:
  - display the current values of the module data,
  - modify/force the data values,
  - diagnose the faulty channels.
- The **Fault** tab, used to display errors occurring in the module.

**NOTE:** The content of these tabs is described in the **"Debugging and adjustment"** chapter of the corresponding application-specific manuals.

Screen example for a counting module:



# Access the Quantum input/output module configuration editor

## Instructions

Proceed with the following steps:

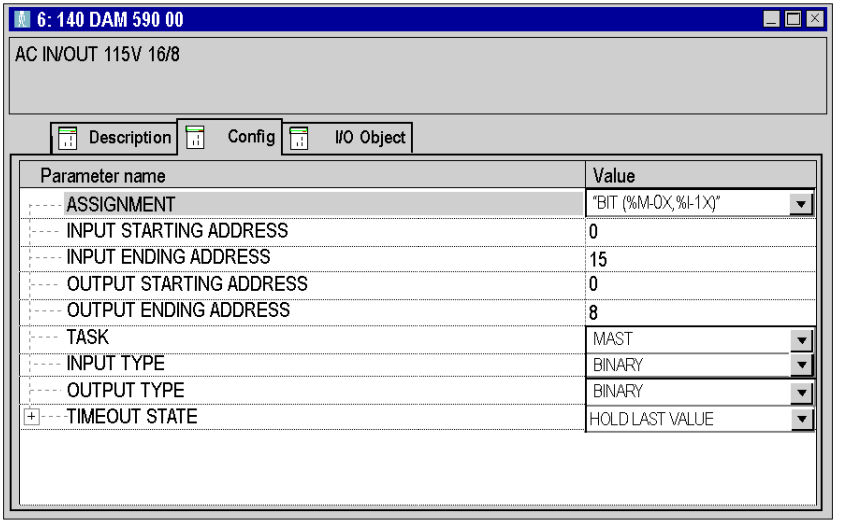
| Step | Action |
|---|---|
| 1 | With the mouse, select the module to be configured. |
| 2 | Using the contextual menu, click **Open Module**. |

## Configuration in offline mode

The configuration window is an editor from which you configure the module, some of your entries are checked, and in some cases you are assisted.

To complete the various fields, **refer to the application-specific manual corresponding to the module** in which you will find the procedure and information necessary for the configuration.

Screen example for a discrete module



### Configuration in online mode

In online mode, for certain modules, two additional tabs are accessible:

- The **Debug** tab, used to:
  - display the current values of the module data,
  - modify/force the data values,
  - diagnose the faulty channels.
- the **Fault** tab which is used to display errors occurring in the module.

**NOTE:** The content of these tabs is described in the corresponding application-specific manuals.

# Chapter 37
# General functionalities of the editor

## Aim of this section

This section describes the general functions available in the configuration editor.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Analysis of the configuration | 1390 |
| Export\Import of the configuration | 1391 |
| Printing the configuration | 1392 |

# Analysis of the configuration

## At a Glance

The configuration analysis is not a separate functionality, it is started during the global analysis of the project.

The configuration analysis checks:

- that all the declared modules are compatible,
- that the maximum number of one type of module in the configuration is not exceeded,
- that the limits of the various types of inputs/outputs managed are not exceeded
- that all the inputs/outputs declared and used in the application are properly configured.

## How to start the global analysis of the application

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the **Build -> Analyse project** command from the menu. |
| 2 | Check the result of the analysis in the information window at the bottom of the screen. |

# Export\Import of the configuration

## Export

The description of the procedure is given in the section "Import/Export *(see page 1731)*".

## Import

The description of the procedure is given in the section "Import/Export *(see page 1732)*".

## Import SIS

This Import SIS command ensures the gateway with the SIS Automation software version greater than or equal to Schneider Electric 3.0.

This import ensures the automatic generation of the configuration: racks, processor, input/output modules... from the .XML file resulting from an export made by this tool.

The description of the procedure is given in the section "Import/Export *(see page 1732)*".

# Printing the configuration

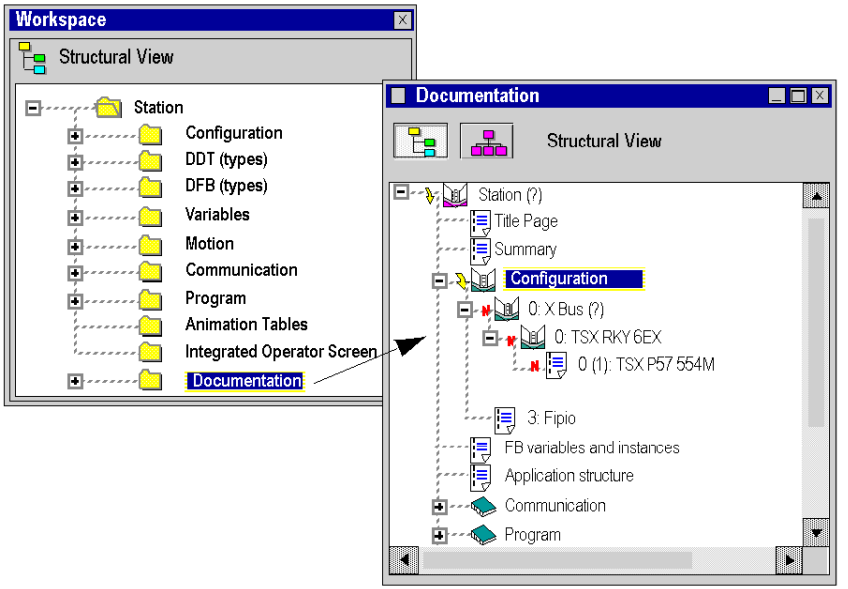
## Introduction

The programming software allows you to generate and print the complete project documentation file. You can choose to print the configuration alone from this file.

In the project documentation file, the configuration part is made up:

- of the bus editor,
- of the input/output module editor.

The documentation file can be accessed from the project browser in the **"Documentation"** directory.




## How to print

A description of the procedure is given in the chapter on "Documentation *(see page 1695)*"

# Part VI
## Debugging and adjustment

### Subject of this Part

This part describes the debug and adjustment functions and tools offered by Unity Pro: application debug, variable adjustment, simulator, etc.

### What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 38 | Debugging the program | 1395 |
| 39 | Viewing and adjusting variables | 1483 |
| 40 | Machine-Process Simulator | 1529 |

# Chapter 38
# Debugging the program

## Subject of this Chapter

This chapter guides the user through how to debug the program depending on the language used in its component sections:

- Ladder language,
- textual languages (Structured Text or Instruction List),
- Function Block Diagram language,
- Sequential Function Chart language.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 38.1 | Safety precautions | 1396 |
| 38.2 | Program Animation | 1406 |
| 38.3 | Debugging in Ladder Language | 1422 |
| 38.4 | Debugging in Textual Languages (Structured Text, Instruction List) | 1431 |
| 38.5 | Debugging in Function Block Diagram (FBD) Language | 1440 |
| 38.6 | Debugging User Function Block (DFB) Instances | 1450 |
| 38.7 | Debugging in Sequential Function Chart Language (SFC) | 1455 |
| 38.8 | Information on Debugging | 1464 |
| 38.9 | PLC debug screen | 1470 |

# Section 38.1
## Safety precautions

### Subject of this Section

This section presents the different safety precautions to respect when debugging a project with Unity Pro.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Modification in RUN | 1397 |
| Switch to STOP of a PLC in Debug mode | 1398 |
| Setting a task to STOP | 1399 |
| Task deactivation | 1400 |
| Deactivation of Inputs/Outputs | 1401 |
| Use of the Step by Step mode and Breakpoint | 1402 |
| Unwanted disconnection in Debug mode for Premium | 1403 |
| Debug Screen Via a Fipway, Fipio or Uni-Telway Network with a PCMCIA Card | 1404 |
| Behavior of Forced Variables between Modsoft/NxT/Concept and Unity Pro | 1405 |

# Modification in RUN

## At a Glance

This functionality is used to modify the PLC program where its inputs/outputs control an industrial process.

## Warning

| ⚠ WARNING |
| --- |
| **UNEXPECTED BEHAVIOUR OF APPLICATION**<br>When using the in Run modification function, ensure that the modifications will not have adverse effects on the running process.<br>Modifications in In Run mode are immediately applied to the process.<br>**Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

## Switch to STOP of a PLC in Debug mode

### At a Glance

You can switch a PLC to STOP using:

- the command **PLC → STOP** when running Unity Pro
- the RUN/STOP input for Modicon M340, Modicon M580, and Premium/Atrium
- the switch on the front panel of the processor for Quantum (except 140 CPU 6••)
- The LCD and buttons for Quantum 140 CPU 6••

### Precaution

In all the above cases, the STOP is only effective at the end of the MAST task cycle.

**⚠ WARNING**

**UNEXPECTED APPLICATION BEHAVIOR**

In Debug mode, check that neither any breakpoint nor the Step by Step mode are active before switching the PLC to STOP.

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

**NOTE:** When the PLC is in STOP, all tasks are in STOP. Check the behavior of the corresponding Inputs/Outputs.

# Setting a task to STOP

## At a Glance

A task can be set to STOP in different ways:

- Use of a breakpoint or the Step by Step mode.
- Use of the PLC screen *(see page 1471)* (PLC animation screen).

## Precautions

**ATTENTION**: When a task is set to STOP, Inputs/Outputs behave differently to whena task is deactivated *(see page 1400)*.

The behavior of outputs differs depending on the type of PLC:

- **Modicon M580, M340 and Premium / Atrium**: when a task is set to STOP, the associated outputs automatically switch to the configured mode (fallback or maintain). The outputs associated with the other tasks continue to be updated as normal.
- **Quantum**: when a task is set to STOP, the associated outputs continue to be updated with the value that preceded the STOP action. The outputs associated with the other tasks continue to be updated as normal.

**NOTE:**
When a task is set to STOP with Quantum, the outputs continue to be updated by the CPU. In two cases, this can lead to unforeseen behavior:

- when inputs are also updated by another task that is still in RUN
- when Inputs/Outputs are mapped in the same memory zone, and associated with different tasks.

**⚠ WARNING**

**UNEXPECTED OUTPUT BEHAVIOR - TASK STOP WITH QUANTUM**

When a task is set to STOP with Quantum, ensure that no other running task is managing the same outputs.

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

# Task deactivation

## At a Glance

A task can be deactivated in different ways:

- By using system bits (%S30 to %S35 *(see Unity Pro, System Bits and Words, Reference Manual)*).
- By using the PLC screen *(see page 1471)* (PLC animation screen).

## Precautions

**ATTENTION**: when a task is deactivated, Inputs/Outputs behave differently to whena task is set to STOP. *(see page 1399)*

The behavior of outputs differs depending on the type of PLC:

| ⚠ WARNING |
| --- |
| **UNEXPECTED APPLICATION BEHAVIOR - TASK DEACTIVATION** |
| When deactivating a task, take into account the fact that Inputs and Outputs continue to be active. |
| **Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

# Deactivation of Inputs/Outputs

## At a Glance

The inputs associated with a task can be deactivated using the system word %SW8 *(see Unity Pro, System Bits and Words, Reference Manual)*.

The outputs associated with a task can be deactivated using the system word %SW9 *(see Unity Pro, System Bits and Words, Reference Manual)*.

## Precautions

**NOTE:** Deactivating the Inputs/Outputs does not deactivate the task *(see page 1400)*

**⚠ WARNING**

**UNEXPECTED APPLICATION BEHAVIOR - OUTPUTS DEACTIVATION**

Before deactivating an Output, take into account its deactivation mode (fallback or maintain).

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

**⚠ CAUTION**

**UNEXPECTED APPLICATION BEHAVIOR**

Before setting the %SW9 value to 1, ensure that the output behavior will remain appropriate:

**On Premium/Atrium:**

Module outputs located on the X Bus automatically switch to the configured mode (fallback or maintain). On the Fipio bus, certain devices do not manage fallback mode; then only maintain mode is possible.

**On Quantum:**

All outputs, as well as the local or remote rack (RIO) are maintained in the state that preceded the switch to 1 of the %SW9 bit corresponding to the task.

The Distributed Inputs/Outputs (DIO) are not assigned by the system word %SW9.

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

## Use of the Step by Step mode and Breakpoint

### At a Glance

The use of the Breakpoint and Step by Step mode influence the PLC's operating modes. You must be careful when using them.

### Precautions

In Debug mode, the use of a breakpoint (except for an SFC breakpoint *(see page 1456)*) or of the Step by Step mode forces the task to STOP.

**⚠ CAUTION**

**UNEXPECTED APPLICATION BEHAVIOR**

Check the consequences on the behavior of the outputs when the task is set to STOP *(see page 1399)*.

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

## Unwanted disconnection in Debug mode for Premium

### Precaution

If the connection is interrupted between the Unity Pro programming software and the Premium PLC, the breakpoint is automatically deleted. If the Premium PLC was stopped at breakpoint level, it automatically switches to STOP after the disconnection Timeout.

**⚠ WARNING**

**UNEXPECTED APPLICATION BEHAVIOR**

With Premium in debug mode, ensure that the consequences of a PLC Stop on the application are acceptable.

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

**NOTE:** When the PLC is in STOP, all tasks are in STOP. Check the behavior of the corresponding Inputs/Outputs.

# Debug Screen Via a Fipway, Fipio or Uni-Telway Network with a PCMCIA Card

## Restriction

If you use debug screens to access a PLC via a Fipway network, a Fipio fieldbus or a Uni-Telway bus with a PCMCIA card, you may, on certain modules, obtain a message explaining that it is not possible to access debugging.

The solution is to connect directly to the PLC with the Uni-Telway, USB or XIP link (depending on the configuration of the hardware used).

# Behavior of Forced Variables between Modsoft/NxT/Concept and Unity Pro

## Forced Variable Behavior

**⚠ WARNING**

**UNEXPECTED VARIABLE BEHAVIOR**

Check your forced variables and memory protection switch when shifting between Modsoft/NxT/Concept and Unity Pro.

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

The behavior of forced variables has been changed between Modsoft/NxT/Concept and Unity Pro.

With Modsoft/NxT/Concept you cannot force variables (0x, 1X) with the Quantum CPU memory protect switch in the ON position.

With Unity Pro you can force variables (%M, %I, %Q) with the Quantum CPU memory protect switch in the ON position.

When using a basic Quantum CPU, if the memory protect switch is in the OFF position, some variables are forced and the application is saved in flash memory:

- with Modsoft/NxT/Concept, forced variables keep their state after a cold start,
- with Unity Pro, forced variables lose their state after a cold start.

## Warning Message when Forcing Variables

A warning message will display in Unity Pro each time that variables are forced on the following Quantum PLCs:

- 140 CPU 311 10
- 140 CPU 434 12A/U
- 140 CPU 534 14A/U

This warning message includes a "Do not warn me again" option so that, when checked, the message will not be redisplayed during the current Unity Pro session. Starting a new session of Unity Pro means that the option must be checked once more in order to no longer see it when forcing a variable.

# Section 38.2
# Program Animation

## Subject of this Section

This section describes the animation used in the different language editors.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction to animation | 1407 |
| Section in textual languages (Structured Text or Instruction List) | 1410 |
| Section in a Ladder Diagram language | 1411 |
| Section in Function Block Diagram (FBD) language | 1414 |
| Section in Sequential Function Chart language (SFC) | 1416 |
| Display of Prepositioned steps, after an on-line modification | 1418 |
| DFB instances (user function blocks) | 1419 |
| Representation of forced variables | 1421 |

# Introduction to animation

## General

Program **animation** is possible if the project that has been opened using the software is identical to the project in the PLC.

The two connection modes are as follows:

- **integral online mode** (projects identical),
- **degraded online mode** (projects different).

In integral online mode, as long as it is authorized for your user profile and the **Use Programming mode** option is checked (**Tools ->Options** menu then **Connection** tab) you can:

- debug the project (setting a breakpoint or a watchpoint),
- modify the project, in this case:
  - either you generate the project, in which case the modifications are automatically transferred to the PLC and animation resumes,
  - or you do not generate the project, in which case animation of the modified sections is stopped.

**NOTE:** Modification and debugging of the project are not permitted in degraded mode, or in integral online mode with the **"Use monitoring mode"** option checked ("Tools ->Options" menu then "Connection" tab).

**NOTE:** In Monitoring mode links between EFs are not animated.

Animation is managed section by section. You can stop it or restart it section by section.

**NOTE:** The screen is refreshed immediately after the execution of each section. As a result, for variables which change very quickly it is possible that there may be differences from time to time between what is displayed and the actual values in the PLC.

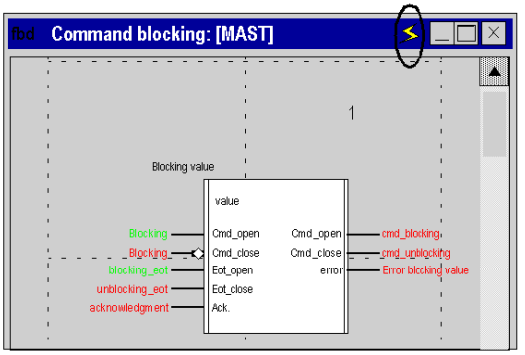
**NOTE:** For all languages, the animation of strings is limited to 16 characters no matter what the string size is. In order to animate more than 16 characters use the animation table with the extended string animation.

## Specific Features of Animation

Two types of animation are available:

- **standard** animation, in which the variables of the active section are refreshed upon completion of the master task (MAST),
- **synchronized** animation in which the variables of the active section are refreshed at the same time as a program element (of the section) containing the **Watchpoint** (useful if one of the variables is used in several program sections and you want to know its value at a specific location)

Example of an animated section with Watchpoint:

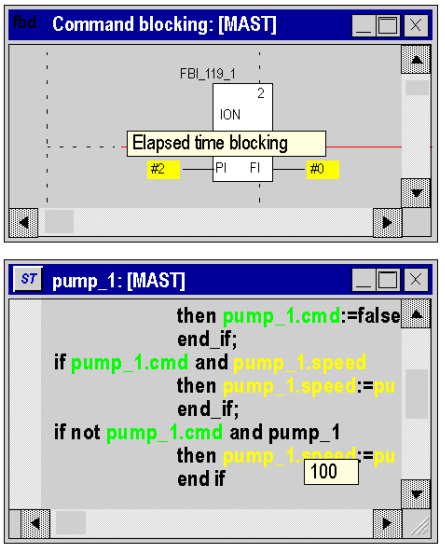



Depending on which language editor is open, it may not be possible to view the name of the variable at the same time as it content. The **Tool tip** function is provided to compensate for this.

This is a help bubble which is displayed when you move the cursor over the variables. This help bubble information is then displayed:

- the value of the variable if only its name is visible in the editor,
- the type, name, address and comment of the variable if only its value is visible in the editor.

Examples of help bubbles in an FBD editor and in a Structured Text editor



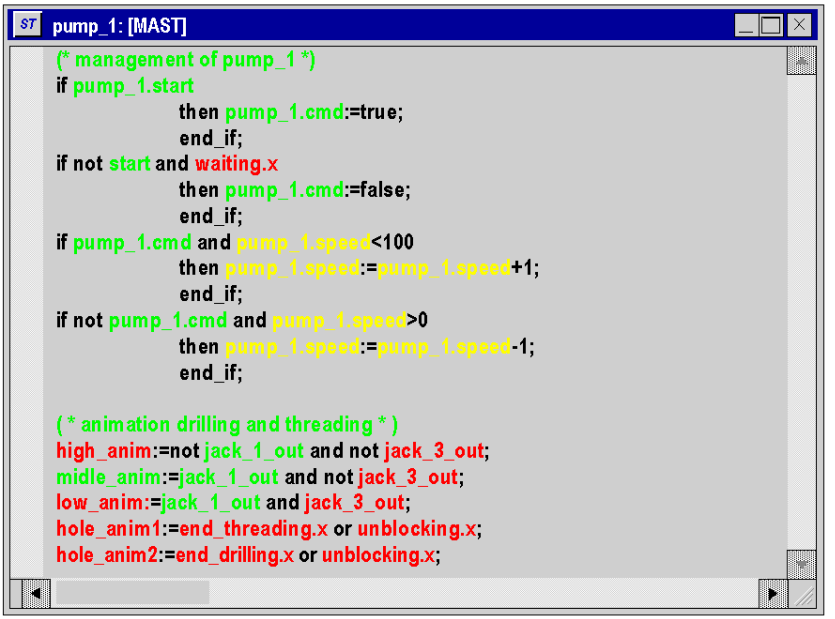
## Starting and Stopping Animation

By default, the sections are animated. To stop the animation of a section click on the  button in the toolbar. To restart the animation, click the button a second time.

## Section in textual languages (Structured Text or Instruction List)

### Animation colors

Animation of a Section in Structured Text language:




The background color of the editor is **gray**.

All text is shown in **black**.

All comments are shown in **green**.

Black color used in offline mode for variables and instructions is replaced.

Variables and instructions during animation are shown in the following colors:

- for Boolean variables:
  - **green** if the variable is TRUE (1)
  - **red** if the variable is FALSE (0)
- **yellow** for most types of variable

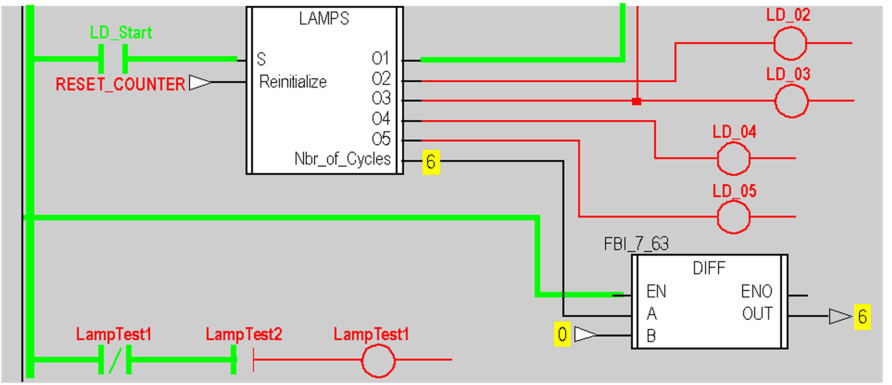
**NOTE:** Data types **other than Booleans**, are animated in a display window and not in the editor. However, it is possible to view their value in a help bubble (Tool tip) when you position the mouse pointer over the value.

**NOTE:** Multi-dimension array variable are not animated in the language editors and are not shown in yellow.

# Section in a Ladder Diagram language

## At a Glance

Animation of a section in Ladder Diagram language:




The background color of the editor is **gray**.

The colors used in offline mode for the various elements (instructions, variables, comments) are replaced:

- for Boolean variables:
  - **green** if the variable is TRUE (1),
  - **red** if the variable is FALSE (0).
- **yellow** for numerical types of variable.

**NOTE:** Derived data **(DDT)** instances and function block **(EFB\DFB)** instances are animated in a display window and not in the language editor. You can view type, name, address and comment of a numerical variable (yellow) in a help bubble (Tool tip) when you position the mouse pointer over the variable.

**NOTE:** In a LD section:
- the unconnected parameters of elementary functions are not animated,
- the unconnected parameters of function blocks are animated.

The result of textual expressions is also animated:

- in green and red for Boolean expressions,
- numerical expressions are replaced by their result, the associated help bubble (Tool tip) displays type, name, address and comment of the expression.

## Animations

There are 3 types of link:

- Boolean links between contacts and coils,
- Boolean links between function blocks,
- numerical links between function blocks.

Two sorts of animation are possible depending on the option selected:

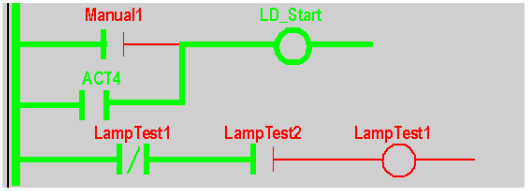
- **with animation of links** for which:
  - Boolean links between contacts and coils are shown in **green** or **red** depending on whether the evaluation of the ladder upstream yields the value TRUE (1) or FALSE (0),
  - Boolean links between function blocks are shown in **green** or **red** depending on whether the output parameter to which they are connected is TRUE (1) or FALSE (0),
  - numerical links between the function blocks are shown in **black**. They display the value of the output parameters to which they are connected.
- **without animation of links** in which closed contacts and triggered coils are shown in reverse video.

To select an animation, perform the following actions:

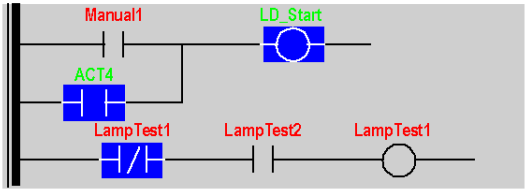
| Step | Action |
|---|---|
| 1 | From the menu, select the **Tools->Project Settings...** command.<br>**Result**: The **Project Settings** *(see page 519)* window is displayed. |
| 2 | In the **Code generation** zone either:<br>● check the **Generate with LD link animation** box for animation of links,<br>● uncheck the **Generate with LD link animation** box to disable animation of links. |
| 3 | Confirm with **OK** |

Example:

With **link animation**




Without link animation

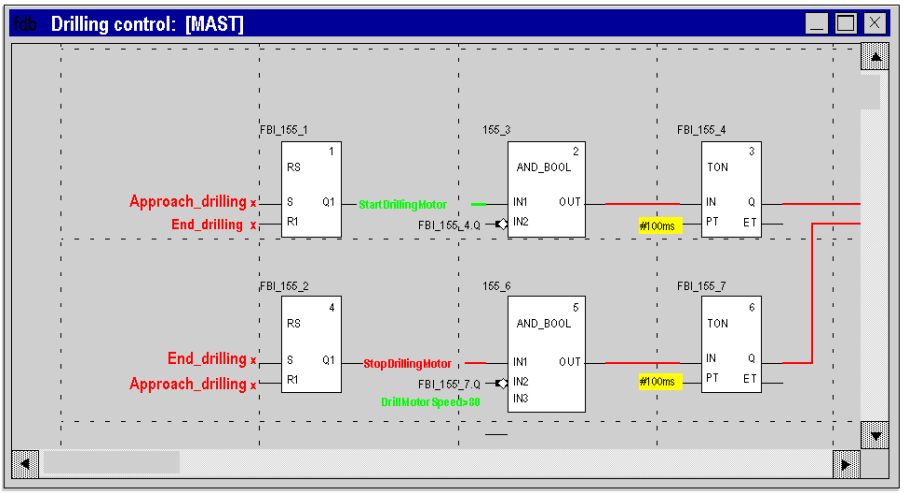



**NOTE:** When switching from one option to the other the project must be regenerated and reloaded into the PLC. Animation of links has an adverse effect on the performance of the project in run.

# Section in Function Block Diagram (FBD) language

## At a Glance

Animation of a section in Function Block Diagram language:




The background color of the editor is **gray**.

The colors used in offline mode for the various elements (instructions, variables, comments) are:

- for Boolean variables:
  - **green** if the variable is TRUE (1),
  - **red** if the variable is FALSE (0).
- **yellow** for numerical types of variable.

**NOTE:** Derived data **(DDT)** instances and function block **(EFB\DFB)** instances are animated in a display window and not in the language editor. You can view type, name, address and comment of a numerical variable (yellow) in a help bubble (Tool tip) when you position the mouse pointer over the variable.

**NOTE:** In an FBD section:
- the unconnected parameters of elementary functions are not animated,
- the unconnected parameters of function blocks are animated.

The result of textual expressions is also animated:

- in green and red for Boolean expressions,
- numerical expressions are replaced by their result, the associated help bubble (Tool tip) displays type, name, address and comment of the expression.

## Animating the Links

There are 2 types of link:

- Boolean links between function blocks,
- numerical links between function blocks.

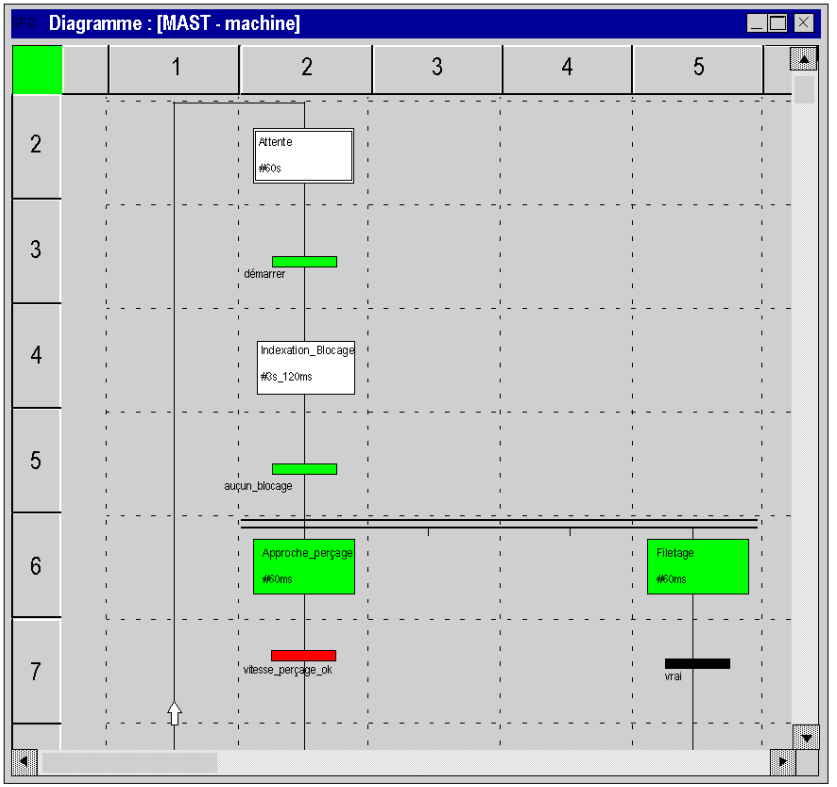
These 2 types of links are animated in the following way:

- Boolean links between function blocks are shown in **green** or **red** depending on whether the output parameters to which they are connected are TRUE (1) or FALSE (0),
- numerical links between the function blocks are shown in **black**. They display the value of the output parameter to which they are connected.

# Section in Sequential Function Chart language (SFC)

## At a Glance

Animation of a section in Sequential Function Chart language:




The background color of the editor is **gray**.

The colors used for the different elements are:

- for steps:
  - **green** if the step is active,
  - **white** if the step is inactive,
  - **yellow** if the activity time of the step is less than the minimum programmed time,
  - **pink** if the activity time of the step is greater than the minimum programmed time.

- for macro-steps:
  - when a macro-step becomes active the upper half is shown in **green**,
  - when the OUT step of the macro-step is active the whole of the macro-step is shown in **green**,
  - when the macro-step becomes inactive it is then shown in **white**.
- for transitions associated with a Boolean element or a simple Boolean expression:
  - **green** if the element or the expression is TRUE
  - **red** if the element or the expression is FALSE.
- for transitions associated with a section:
  - **black** as long as the previous step remains inactive,
  - **green** if the conditions in the section are TRUE,
  - **red** if the conditions in the section are FALSE,

| ⚠ CAUTION |
| --- |
| **ANIMATION OF SFC EDITOR STOPPED**<br>In online mode with the animation activated, you can modify the program and Generate the project. In this case, the animation is not automatically restarted.<br>To return to the animation, click **Services → Animation**<br>**Failure to follow these instructions can result in injury or equipment damage.** |

# Display of Prepositioned steps, after an on-line modification

## At a Glance

As part of the on-line modifications associated with a prepositioning of the SFC, Unity Pro behaves in a specific way when:

- An SFC section uses macro sections
- Sections are animated.
- A prepositioning is performed for this section and for the sections of the macro-steps.

## Procedure

Procedure for obtaining the scenario that may cause a problem.

| Step | Action |
|---|---|
| 1 | Perform a modification in one of the sections.<br>Result:<br>● The animation of the section is stopped.<br>● The prepositionings of the SFC and its macro-sections are reset to zero. |
| 2 | Use the command **Build** → **Build Changes** to generate the modifications. |
| 3 | Use the command **PLC** → **Transfer project to PLC** to load the modifications in the PLC.<br>Result:<br>● The animation of the modified section is relaunched and displays the current state (the prepositioning is reset to zero).<br>● The animation of the sections belonging to the modified section is not updated. In fact, these still display the prepositionings despite the fact they have been deleted. |

## Solution:

Stop the animation of the sections concerned, then relaunch this animation again.

# DFB instances (user function blocks)

## At a Glance

A DFB instance may be made up of one section (IEC 1131-3) or several sections.

Animation of a DFB instance consists in **animating a section of the instance** written in one of the programming languages.

The rules governing animation of the DFB sections are the same as for conventional sections according to the language used such as:

- textual languages *(see page 1410)*,
- Ladder Diagram language *(see page 1411)*,
- Function Block Diagram language *(see page 1414)*.

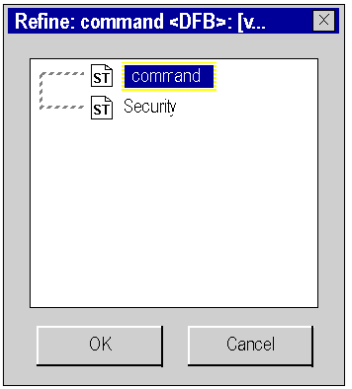
The animated variables are the variables belonging to the DFB instance that has a section which is currently being animated, in other words:

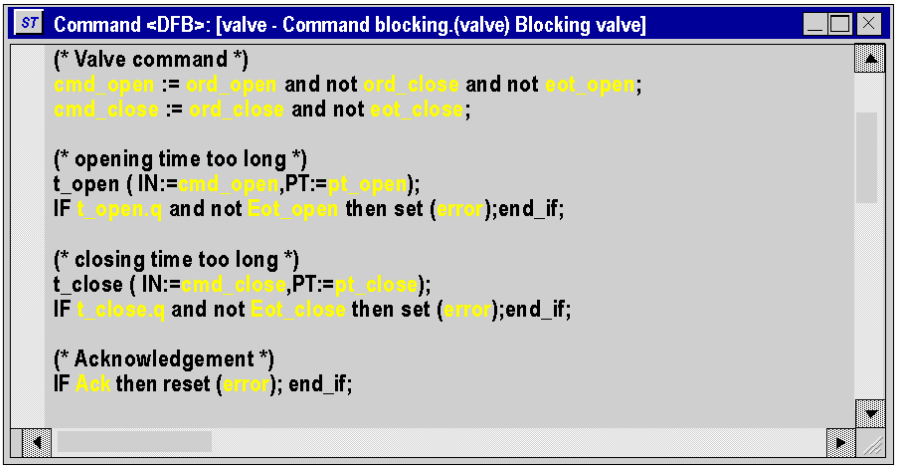
- the input, input\output, output parameters,
- the public variables,
- the private variables.

**NOTE:** in a DFB section, the unconnected parameters of function blocks are not animated.

## How to Animate a Section of a DFB

To go to the relevant section of a DFB instance, perform the following operations:

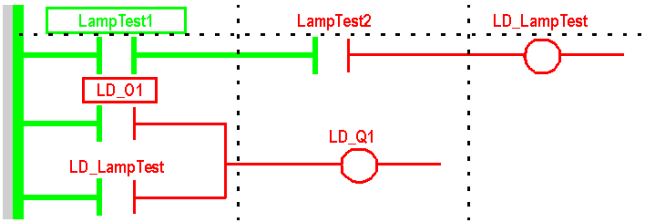
| Step | Action |
|---|---|
| 1 | In the **animated** section of the program, select the DFB instance and from the contextual menu execute the **Refine** command.<br>A window showing the DFB section(s) opens:<br>  |

| Step | Action |
|---|---|
| 2 | In this box, choose the section of the DFB that you wish to animate and confirm with **OK**.<br>Section in Structured Text:<br>  <br> |
| 3 | If the DFB section which is currently animated is a **nested instance** and you wish to animate one of its sections, restart the operation. |

# Representation of forced variables

## At a Glance

Animation of forced variables:




**NOTE:** Forcing only concerns located Boolean variables of EBOOL type.

**NOTE:** For Modicon M580 and M340 PLCs analog I/O variables can also be forced.

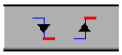
In **animated mode**, these variables are represented in the following way:

- framed by a **green** rectangle if the value is TRUE (1),
- framed by a **red** rectangle if the value is FALSE (0).

**NOTE:** Analog I/O variables don't have any special representation when forced. (They are not framed by rectangles)

## Implementation

The table below describes the actions to be performed to force variables from a language editor:

| Step | Action |
|---|---|
| 1 | Open a section. |
| 2 | From the **Edit** menu, select all variables using the **Select all** command or<br>Select a variable. |
| 3 | Select the **Initialize Animation Table** command from the **Services** menu.<br>**Result**: An animation table opens with the variable(s) selected in the section. |
| 4 | Click on **Force** in the animation table. |
| 5 | Select the variable to be forced in the animation table. |
| 6 | In the animation table click on one of the  buttons alongside the desired value or execute the **Force to 0** or **Force to 1** commands from the contextual menu. |

# Section 38.3
# Debugging in Ladder Language

## Subject of this Section

This section introduces the different tools available and explains how to implement them to debug the program. These tools are:

- the breakpoint,
- step by step execution,
- the watchpoint.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Inserting a breakpoint in Ladder Diagram language (LD) | 1423 |
| Step by step execution mode in Ladder Diagram language (LD) | 1425 |
| Inserting a Watchpoint in Ladder Diagram Language | 1428 |

# Inserting a breakpoint in Ladder Diagram language (LD)

## At a Glance

A breakpoint can be used to **stop the execution** of the task at the point at which it was set.

During debugging it can be used:

- to examine the behavior of the code,
- to view the value of variables.

There is a single breakpoint at a given point in the project. This is not saved and is lost on disconnection from the PLC.

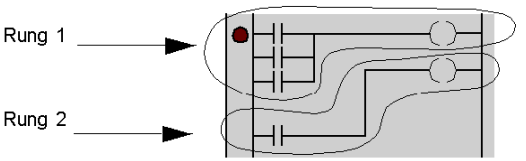
It is implemented in online mode, regardless of whether the PLC is in **Run** or in **Stop**.

**NOTE:** It is not possible to set a breakpoint in an event task.

## Element Used for Setting a Breakpoint

The program element which can be used to set a breakpoint is the **rung** or a Sub-Routine (SR) or user function block (DFB) call within a rung.

In other words, the following program section:

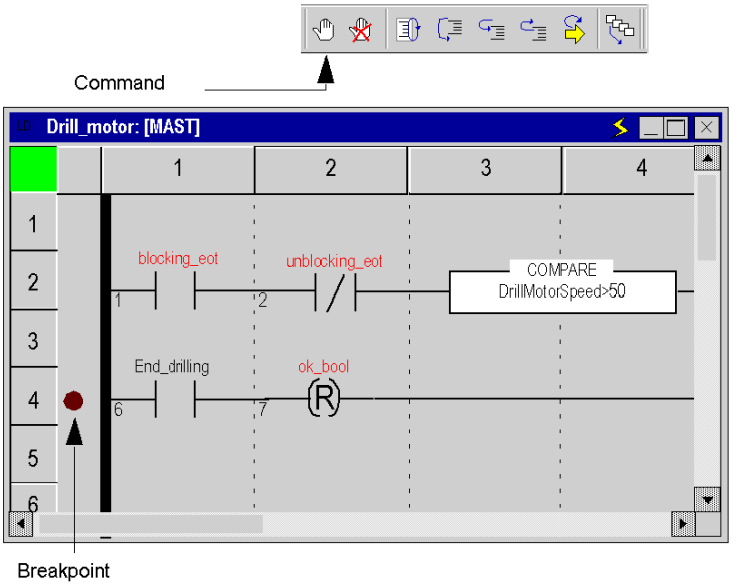


## How to Insert a Breakpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | For example, select a contact in the rung. |
| 2 | Set the breakpoint:<br>● by selecting the **Debug->Set Breakpoint** command from the menu,<br>● by selecting the **Set Breakpoint** command from the contextual menu,<br>● or by selecting the button in the debug toolbar. |

Example:




**NOTE:** Inserting a new breakpoint automatically clears the old one.

From the menu, select the **Debug->Show Breakpoint** command to locate the breakpoint by displaying the section where it has been set in the language editor.

## How to Delete a Breakpoint

Either you can:

- select the **Debug->Clear Breakpoint** command from the menu,
- or select the button in the debug toolbar.

Erasing a breakpoint does not restart the task. To do this, you have to press .

# Step by step execution mode in Ladder Diagram language (LD)

## At a Glance

Step by step mode consists in executing the application program **rung by rung**. For rungs containing sub-routine (SR) or user function block (DFB) calls, step by step can be written into the code used to implement them.

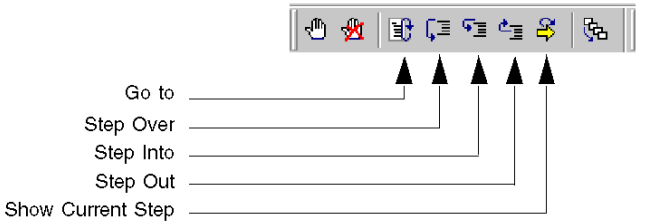
This mode is launched by a **breakpoint** which will have been set in advance. It is used to examine the behavior of the code, and the value of the variables.

It is implemented in online mode. The section executed in step by step mode stops the corresponding task, and the links are no longer animated.

Three commands can be used for step by step mode. These are:

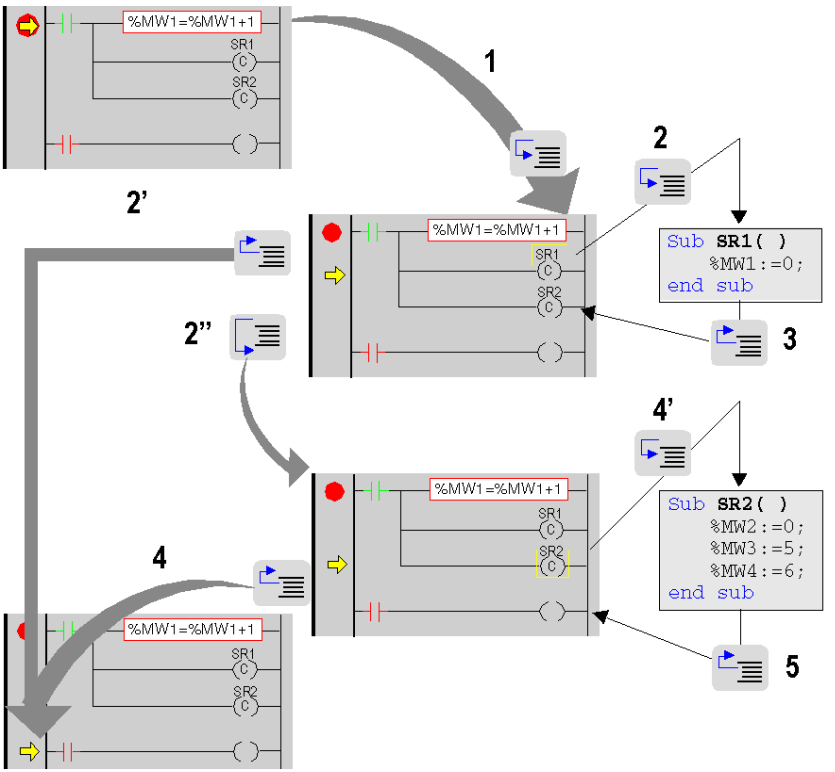
- the **Step Into** command,
  - if the current element is a rung which does not contain an SR call or a DFB instance call, this command executes the rung and moves onto the next rung,
  - if the current element is a rung which does contain an SR call or a DFB instance call, this command executes the start of the rung and moves to the first SR or DFB call.
  - if the current element is an SR or DFB instance call, this command can be used to step into the code and go to the first element.
- the **Step Over** command,
  - if the current element is a rung, this command executes it in its entirety and moves onto the next rung,
  - if the current element is an SR or DFB instance call, this command executes it in its entirety as if it were a straightforward element and moves onto the next element.
- the **Step Out** command.
  - if the current element is part of the code of an SR or DFB instance, this command can be used to execute all the elements of the SR or the DFB and to move onto the next element after the SR or DFB call,
  - if the current element is an SR call or a DFB instance call, this command executes the remainder of the current rung in its entirety and moves onto the next rung,
  - if the current element is a rung, this command executes the current section in its entirety and moves onto the start of the next section.

From step by step mode, if you want to relaunch the execution of the task in order to return to the **breakpoint** you set earlier, select the **Debug->Go** command from the menu, or click on **Go** in the toolbar:




## Step Into, Step Out and Step Over

Example:



Description of step by step from the breakpoint reached  :

| Step | Action |
|---|---|
| 1 | Clicking on the **Step Into** button moves execution of the program to the SR1 call. |
| 2 or 2' or 2'' | If you click on:<br>● **Step Into** (2) execution of the program moves onto `%MW1:=0`<br>● **Step Out** (2') the rung is executed and execution of the program moves onto the next rung.<br>● **Step Over** (2'') SR1 is executed and execution of the program moves on to call SR2. |
| 3 | Clicking on **Step Out** while in SR1 executes SR1 in its entirety and moves execution of the program on to call SR2. |
| 4 or 4' or | If you click on:<br>● **Step Into** (4) execution of the program moves onto `%MW2:=0`<br>Clicking on **Step Into** moves execution of the program onto `%MW3:=5`, and so on if you want to use step by step in SR2.<br>● **Step Out** (4) the rung is executed and execution of the program moves onto the next rung, |
| 5 | Clicking on **Step Out** while in SR2 executes SR2 in its entirety and moves execution of the program onto the next rung. |

**NOTE:** When program execution is over an instruction, it will not yet have been executed. It will be executed after a command is pressed.

### Specific Case of Step Out

If you want to perform step by step **section by section**, execute the **Step Out** command from the first element of each section.

**NOTE:** If the next section is an **SFC section or a section implementing an action or a transition** (mandatory in the Mast task), this is skipped unless a language element has a breakpoint.

### Rule

In step by step mode the task manager will not detect a **watchdog overflow** for the task currently being debugged. It will, however, detect infinite loops.

# Inserting a Watchpoint in Ladder Diagram Language

## At a Glance

When there is no watchpoint, the values of animated variables are displayed at the end of MAST task processing.

The limitation of this mode of operation is that it does not permit the value of a variable at a given point in the program to be known if this value is used in different sections.

The watchpoint is used to **synchronize display of animated variables** with **execution of a program element** (rung) in order to know their exact value at this specific point in the program.

Variables which are synchronous with the watchpoint **must belong to the section** in which the watchpoint is set; display of other variables is synchronous with the end of the MAST task.

These variables are displayed:

- in the section ladder editor,
- in the display window *(see page 1485)*,
- in a single animation table at a given moment when requested by the user.

**NOTE:** It is not possible to set a watchpoint in an event task.

## Watchpoint Properties

These properties are:

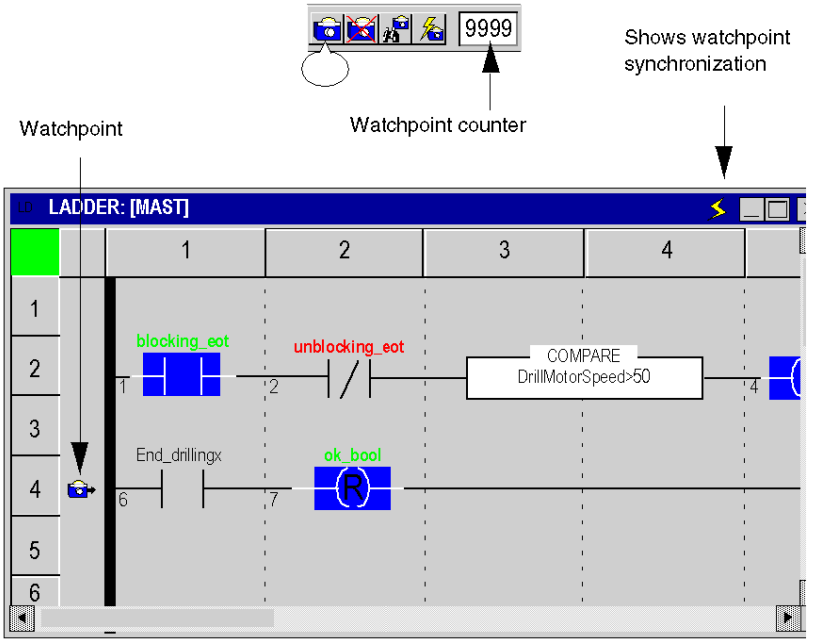
- the watchpoint can be set only in online mode; if the connection is broken, the watchpoint is lost as well,
- a single watchpoint is allowed at a given moment; the watchpoint is exclusive with the breakpoint,
- display of animated variables is effective before execution of the rung on which the watchpoint has been set,
- a counter is incremented each time the rung with the watchpoint is executed, up to a maximum value of 9999, then reset to 0,
- modification of a section is not authorized if a watchpoint is set.

## How to Insert a Watchpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | For example, select a contact in the rung. |
| 2 | Set the watchpoint in one of the following ways:<br>● by selecting the following command from the menu: **Debug->Set Watchpoint**,<br>● by selecting the following command from the contextual menu: **Set Watchpoint**,<br>● or select the  button in the toolbar. |

Example:




**NOTE:** Inserting a new watchpoint automatically clears the old one.

## How to Locate an Existing Watchpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Either:<br>● by selecting the following command from the menu:<br>**Debug->Show Watchpoint,**<br>● or select the  button in the toolbar. |
| 2 | The part of the language editor containing the watchpoint is displayed. |

## How to Synchronize with an Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the animation table desired. |
| 2 | either you can:<br>● select the following command from the menu:<br>**Debug->Synchronize Animation Table**.<br>● or select the  button in the toolbar. |

## How to Delete a Watchpoint

Either you can:

- select the **Debug->Clear Watchpoint** command from the menu,
- or select the  button in the toolbar.

# Section 38.4
## Debugging in Textual Languages (Structured Text, Instruction List)

### Subject of this Section

This section introduces the different tools available and explains how to implement them to debug the program. These tools are:

- the breakpoint,
- step by step execution,
- the watchpoint.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Insertion of a breakpoint in textual languages (Structured Text or Instruction List) | 1432 |
| Step by step execution mode in textual languages (Structured Text or Instruction List) | 1434 |
| Insertion of a Watchpoint in Textual Languages (Structured Text or Instruction List) | 1437 |

# Insertion of a breakpoint in textual languages (Structured Text or Instruction List)

## At a Glance

A breakpoint can be used to **stop the execution** of the task at the point at which it was set.

During debugging it can be used:

- to examine the behavior of the code,
- to view the value of variables.

There is a single breakpoint at a given point in the project. This is not saved and is lost on disconnection from the PLC.

It is implemented in online mode regardless of whether the PLC is in **Run** or in **Stop**.

**NOTE:** It is not possible to set a breakpoint in an event task.

## Element Used for Setting a Breakpoint

The program element used to set a breakpoint is an **instruction**.

In other words, the following program section:

```
Line 1 (*Example*)
Line 2 IF(%MW10=14) THEN  (*Test condition*)
Line 3     %MW45:=68;     (*Execution if condition true*)
Line 4 ELSE
Line 5     %MW45:=24;     (*Execution if condition false*)
Line 6 END_IF;
Line 7 %MW10:=12; DFB1_3(prms); %MW0:=%MW0+1;
Line 8
```

Lines 2, 3, 5 and 7 can hold a breakpoint.

Lines 1, 4, 6 and 8 cannot hold a breakpoint.

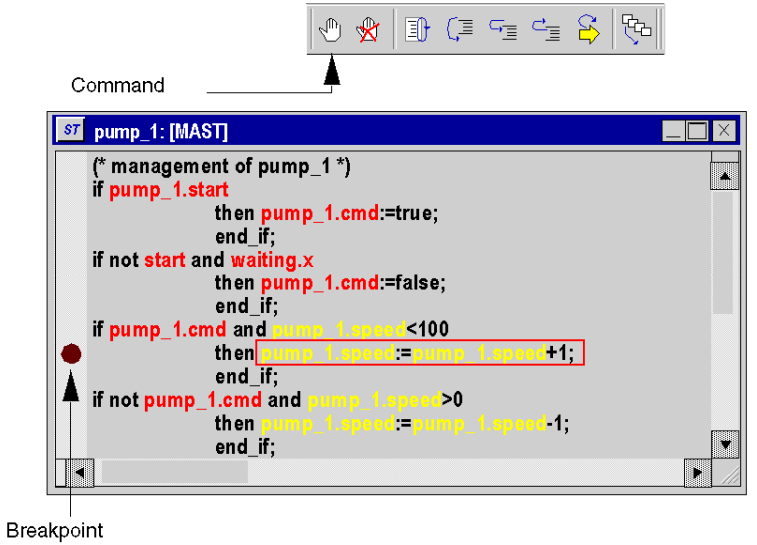
**NOTE:** On line 7 (several instructions), the breakpoint can be set on the first, second or third instruction. **The instruction with the breakpoint is framed.**

## How to Insert a Breakpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the desired program element. |
| 2 | Set the breakpoint:<br>● by selecting the following command from the menu: **Debug->Set Breakpoint**,<br>● or by selecting the  button in the debug toolbar. |

Example:




**NOTE:** Inserting a new breakpoint automatically clears the old one.

From the menu, select the **Debug->Show Breakpoint** command to locate the breakpoint by displaying the section where it has been set in the language editor.

## How to Delete a Breakpoint

Either you can:

- select the **Debug->Clear Breakpoint** command from the menu,
- or select the  button in the debug toolbar.

Erasing a breakpoint does not restart the task. To do this, you have to press  .

# Step by step execution mode in textual languages (Structured Text or Instruction List)

## At a Glance

Step by step mode consists in executing the application program **instruction by instruction**. A line may contain several instructions.

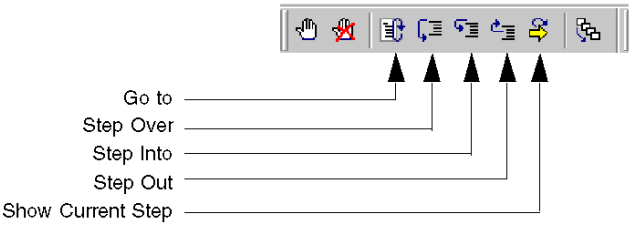
This mode is launched by a **breakpoint** which will have been set in advance. It is used to examine the behavior of the code, and the value of the variables.

It is implemented in online mode. The section executed in step by step mode stops the corresponding task.

Three commands can be used for step by step mode. These are:

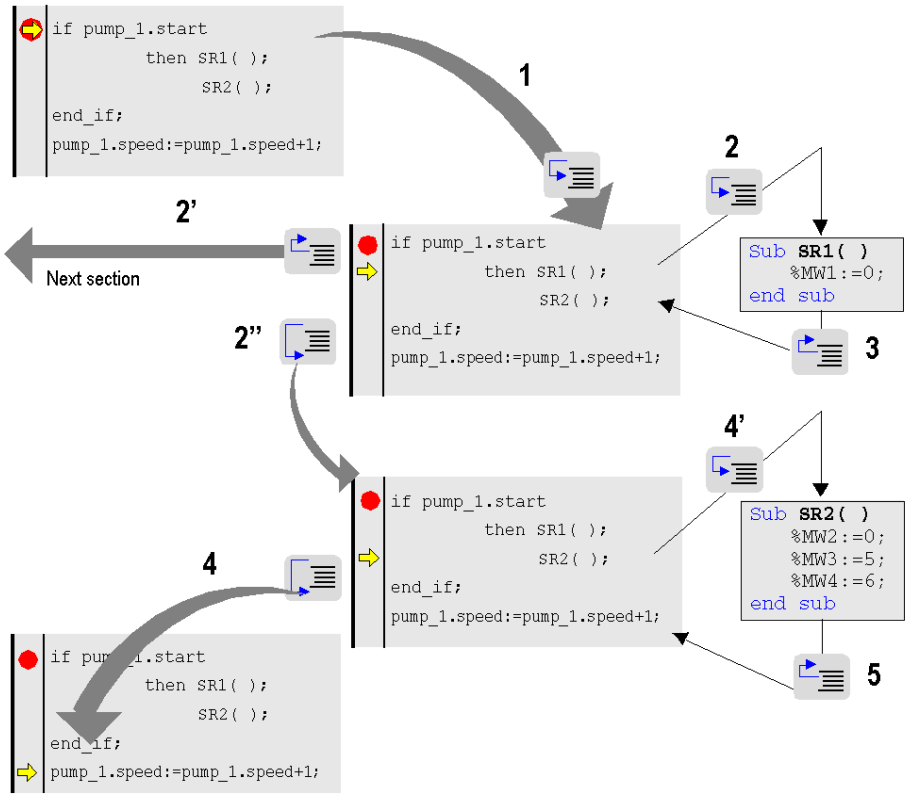
- the **Step Into** command,
  - if the current element is an SR (sub-routine) or DFB (user function block) instance call, this command can be used to step into the code and go to the first element of the SR or DFB,
  - if the current element is an instruction, this command executes it and moves onto the next instruction.
- the **Step Over** command,
  - if the current element is an SR or DFB instance call, this command executes it in its entirety as if it were a straightforward element and moves onto the next instruction,
  - if the current element is an instruction, this command executes it and moves onto the next instruction.
- the **Step Out** command.
  - if the current element is part of the code of an SR or DFB, this command can be used to execute all the elements of the SR or the DFB and to move onto the next element of the SR or DFB.
  - if the current element is an instruction, this command executes the current section in its entirety and moves onto the start of the next section.

From step by step mode, if you want to relaunch the execution of the task in order to return to the **breakpoint** you set earlier, select the **Debug->Go** command from the menu, or click on **Go** in the toolbar:



## Step Into, Step Out and Step Over

Example:




Description of step by step from the breakpoint reached  :

| Step | Action |
|---|---|
| 1 | Clicking on the **Step Into** button moves execution of the program to the SR1 call. |
| 2 or 2' or 2'' | If you click on:<br>● **Step Into** (2) execution of the program moves onto `%MW1:=0`<br>● **Step Out** (2') the section is executed and execution of the program moves onto the first element of the next section.<br>● **Step Over** (2'') SR1 is executed and execution of the program moves on to call SR2. |

| Step | Action |
|---|---|
| 3 | Clicking on **Step Out** while in SR1 executes SR1 in its entirety and moves execution of the program on to call SR2. |
| 4 or 4' | If you click on:<br>● **Step Into** (4) execution of the program moves onto `%MW2:=0`<br>Clicking on **Step Into** moves execution of the program onto `%MW3:=5`, and so on if you want to use step by step in SR2.<br>● **Step Over** (4) SR2 is executed and execution of the program moves onto the next instruction. |
| 5 | Clicking on **Step Out** while in SR2 executes SR2 in its entirety and moves execution of the program onto the next instruction. |

**NOTE:** When program execution is over an instruction, it will not yet have been executed. It will be executed after a command is pressed.

### Specific Case of Step Out

If you want to perform step by step **section by section**, execute the **Step Out** command from the first element of each section.

**NOTE:** If the next section is an **SFC section or a section implementing an action or a transition** (mandatory in the Mast task), this is skipped unless a language element has a breakpoint.

### Rule

In step by step mode the task manager will not detect a **watchdog overflow** for the task currently being debugged. It will, however, detect infinite loops.

# Insertion of a Watchpoint in Textual Languages (Structured Text or Instruction List)

## At a Glance

When there is no watchpoint, the values of animated variables are displayed at the end of MAST task processing.

The limitation of this mode of operation is that it does not permit the value of a variable at a given point in the program to be known if this value is used in different sections.

The watchpoint is used to **synchronize display of animated variables** with **execution of a program element** (instruction) in order to know their exact value at this specific point in the program.

Variables which are synchronous with the watchpoint must belong to the section in which the watchpoint is set; display of other variables is synchronous with the end of the MAST task.

These variables are displayed:

- in the section language editor,
- in the display window,
- in a single animation table at a given moment when requested by the user.

**NOTE:** It is not possible to set a watchpoint in an event task.

**NOTE:** Using a watchpoint degrades real-time performances due to increased application overhead. It is not recommended to use a watchpoint in a code loop, because the watchpoint would be called n times in a cycle and cause major overhead, possibly triggering the task watchdog.

## Watchpoint Properties

These properties are:

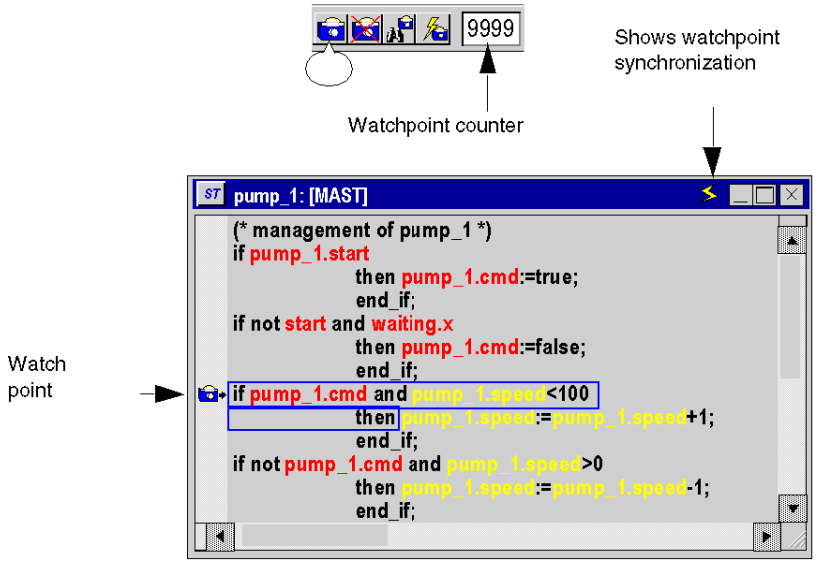
- the watchpoint can be set only in online mode; if the connection is broken, the watchpoint is lost as well,
- a single watchpoint is allowed at a given moment; the watchpoint is exclusive with the breakpoint,
- display of animated variables is effective before execution of the instruction on which the watchpoint has been set,
- a counter is incremented each time the instruction with the watchpoint is executed, up to a maximum value of 9999, then reset to 0,
- modification of a section is not authorized if a watchpoint is set.

## How to Insert a Watchpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the section, select the desired instruction. |
| 2 | Set the watchpoint in one of the following ways:<br>● by selecting the following command from the menu: **Debug->Set Watchpoint**,<br>● by selecting the **Set Watchpoint** command from the contextual menu,<br>● or select the  button in the toolbar. |

Example:




**NOTE:** Inserting a new watchpoint automatically clears the old one.

## How to Locate an Existing Watchpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Either:<br>● by selecting the following command from the menu:<br>**Debug->Show Watchpoint**.<br>● or select the  button in the toolbar. |
| 2 | The part of the language editor containing the watchpoint is displayed. |

## How to Synchronize with an Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the animation table desired. |
| 2 | either you can:<br>● select the following command from the menu:<br>**Debug->Synchronize Animation Table**.<br>● or select the  button in the toolbar. |

## How to Delete a Watchpoint

Either you can:

- select the following command from the menu:
  **Debug->Clear Watchpoint**,
- or select the  button in the toolbar.

# Section 38.5
# Debugging in Function Block Diagram (FBD) Language

## Subject of this Section

This section introduces the different tools available and explains how to implement them to debug the program. These tools are:

- the breakpoint,
- step by step execution,
- the watchpoint.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Inserting a breakpoint in Function Block Diagram language | 1441 |
| Step by step execution mode in Function Block Diagram language (FBD) | 1444 |
| Inserting a watchpoint in Function Block Diagram language (FBD) | 1447 |

# Inserting a breakpoint in Function Block Diagram language

## At a Glance

A breakpoint can be used to **stop the execution** of the task at the point at which it was set.

During debugging it can be used:

- to examine the behavior of the code,
- to view the value of variables.

There is a single breakpoint at a given point in the project. This is not saved and is lost on disconnection from the PLC.

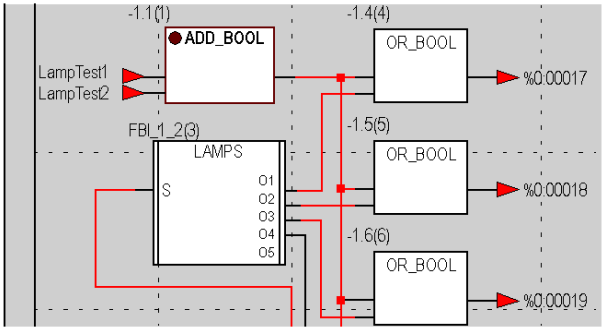
It is implemented in online mode regardless of whether the PLC is in **Run** or in **Stop**.

**NOTE:** It is not possible to set a breakpoint in an event task.

## Element Used for Setting a Breakpoint

The program element used to set a breakpoint is the **function block**.

In other words, the following program section:



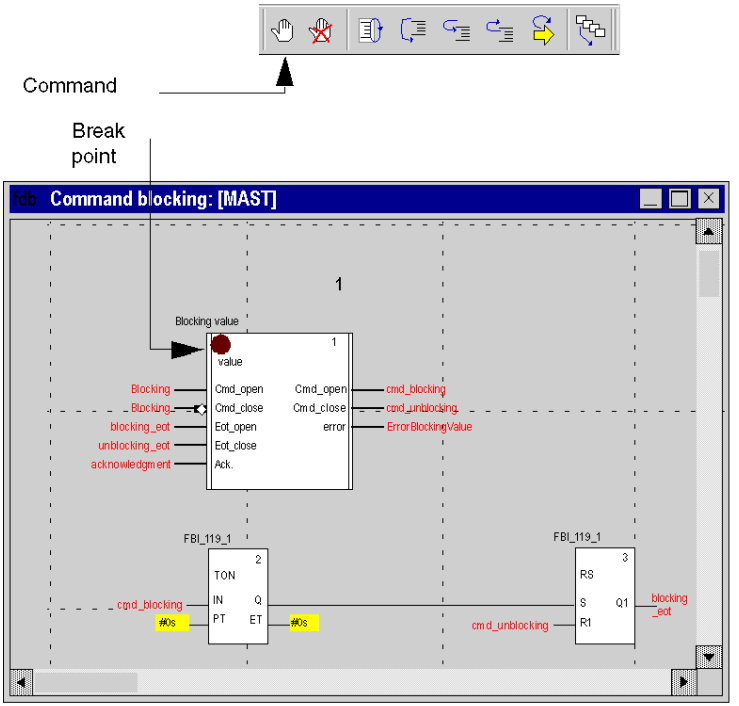

The breakpoint can only be set on 1 block at a time. The number in brackets over the block corresponds to the order of execution.

## How to Insert a Breakpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the desired program element. |
| 2 | Set the breakpoint:<br>● by selecting the following command from the menu: **Debug->Set Breakpoint**,<br>● by selecting the following command from the contextual menu: **Set Breakpoint**,<br>● or by selecting the  button in the debug toolbar. |

Example:




**NOTE:** Inserting a new breakpoint automatically clears the old one.

From the menu, select the **Debug->Show Breakpoint** command to locate the breakpoint by displaying the section where it has been set in the language editor.

### How to Delete a Breakpoint

Either you can:

- select the following command from the menu:
  **Debug->Clear Breakpoint**,
- or select the  button in the debug toolbar.

Erasing a breakpoint does not restart the task. To do this, you have to press  .

# Step by step execution mode in Function Block Diagram language (FBD)

## At a Glance

Step by step mode consists in executing the application program **function block by function block**.

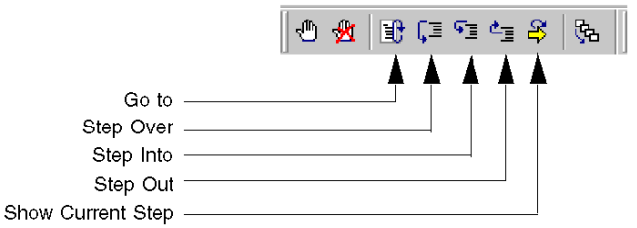
This mode is launched by a **breakpoint** which will have been set in advance. It is used to examine the behavior of the code, and the value of the variables.

It is implemented in online mode. The section executed in step by step mode stops the corresponding task.

Three commands can be used for step by step mode. These are: :

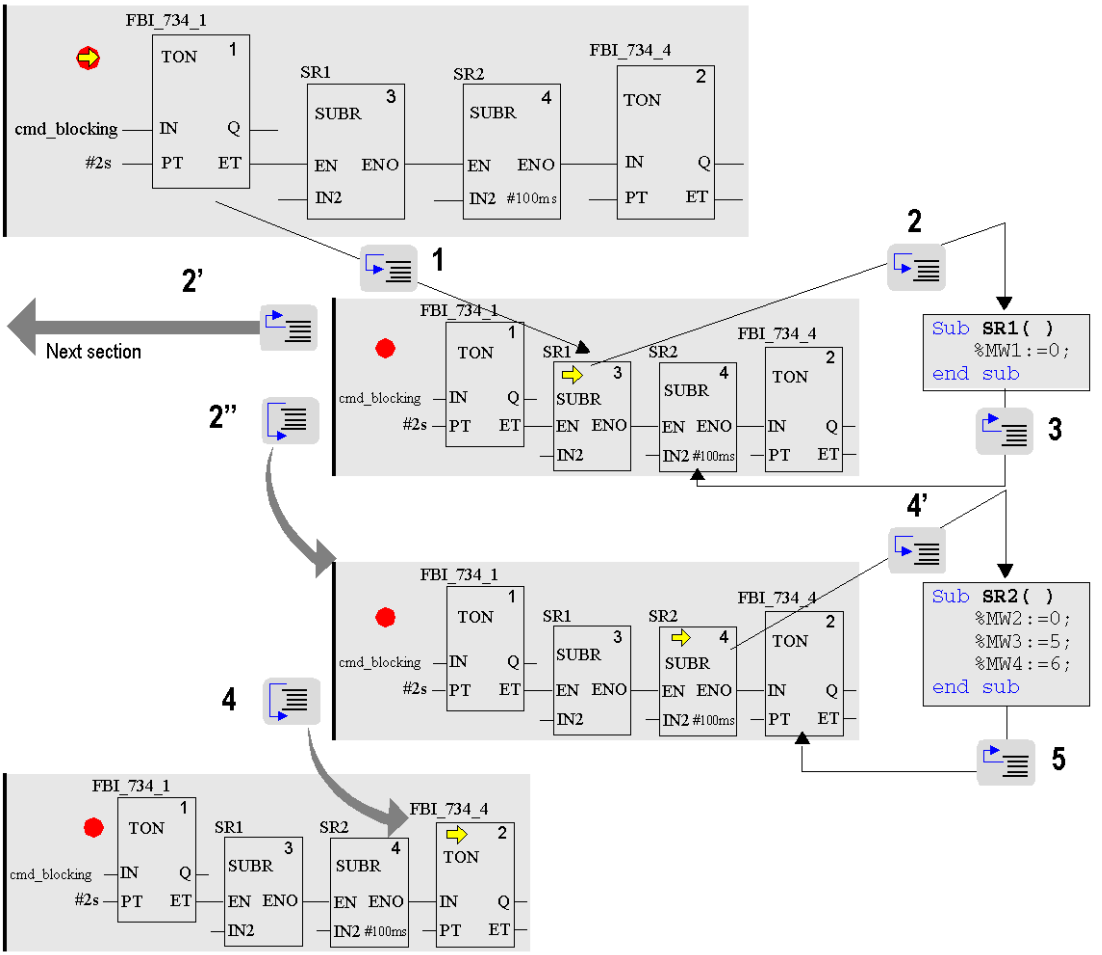
- the **Step Into** command,
  - if the current element is an SR (sub-routine) or DFB (user function block) instance call, this command can be used to step into the code and go to the first element of the SR or DFB,
  - if the current element is a function block, this command executes it and moves onto the next function block.
- the **Step Over** command,
  - if the current element is an SR or DFB instance call, this command executes it in its entirety as if it were a straightforward element and moves onto the next function block,
  - if the current element is a function block, this command executes it and moves onto the next function block.
- the **Step Out** command.
  - if the current element is part of the code of an SR or DFB, this command can be used to execute all the elements of the SR or the DFB and to move onto the next element of the SR or DFB.
  - if the current element is a function block, this command executes the current section in its entirety and moves onto the start of the next section.

From step by step mode, if you want to relaunch the execution of the task in order to return to the **breakpoint** you set earlier, select the **Debug->Go** command from the menu, or click on **Go** in the toolbar:



## Step Into, Step Out and Step Over

Example:



Description of step by step from the breakpoint reached  :

| Step | Action |
|---|---|
| 1 | Clicking on the **Step Into** button moves execution of the program to the SR1 call. |
| 2 or 2' or 2'' | If you click on:<br>● **Step Into** (2) execution of the program moves onto `%MW1:=0`<br>● **Step Out** (2') the section is executed and execution of the program moves onto the first element of the next section.<br>● **Step Over** (2'') SR1 is executed and execution of the program moves on to call SR2. |
| 3 | Clicking on **Step Out** while in SR1 executes SR1 in its entirety and moves execution of the program on to call SR2. |
| 4 or 4' | If you click on:<br>● **Step Into** (4) execution of the program moves onto `%MW2:=0`<br>Clicking on **Step Into** moves execution of the program onto `%MW3:=5`, and so on if you want to use step by step in SR2.<br>● **Step Over** (4) SR2 is executed and execution of the program moves onto the next function block. |
| 5 | Clicking on **Step Out** while in SR2 executes SR2 in its entirety and moves execution of the program onto the next function block. |

**NOTE:** When program execution is over a function block, it will not yet have been executed. It will be executed after a command is pressed.

### Specific Case of Step Out

If you want to perform step by step **section by section**, execute the **Step Out** command from the first element of each section.

**NOTE:** If the next section is an **SFC section or a section implementing an action or a transition** (mandatory in the Mast task), this is skipped unless a language element has a breakpoint.

### Rule

In step by step mode the task manager will not detect a **watchdog overflow** for the task currently being debugged. It will, however, detect infinite loops.

# Inserting a watchpoint in Function Block Diagram language (FBD)

## At a Glance

When there is no watchpoint, the values of animated variables are displayed at the end of MAST task processing.

The limitation of this mode of operation is that it does not permit the value of a variable at a given point in the program to be known if this value is used in different sections.

The watchpoint is used to **synchronize display of animated variables** with **execution of a program element** (function block) in order to know their exact value at this specific point in the program.

Variables which are synchronous with the watchpoint **must belong to the section** in which the watchpoint is set; display of other variables is synchronous with the end of the MAST task.

These variables are displayed:

- in the section language editor,
- in the display window,
- in a single animation table at a given moment when requested by the user.

**NOTE:** It is not possible to set a watchpoint in an event task.

## Watchpoint Properties

These properties are:

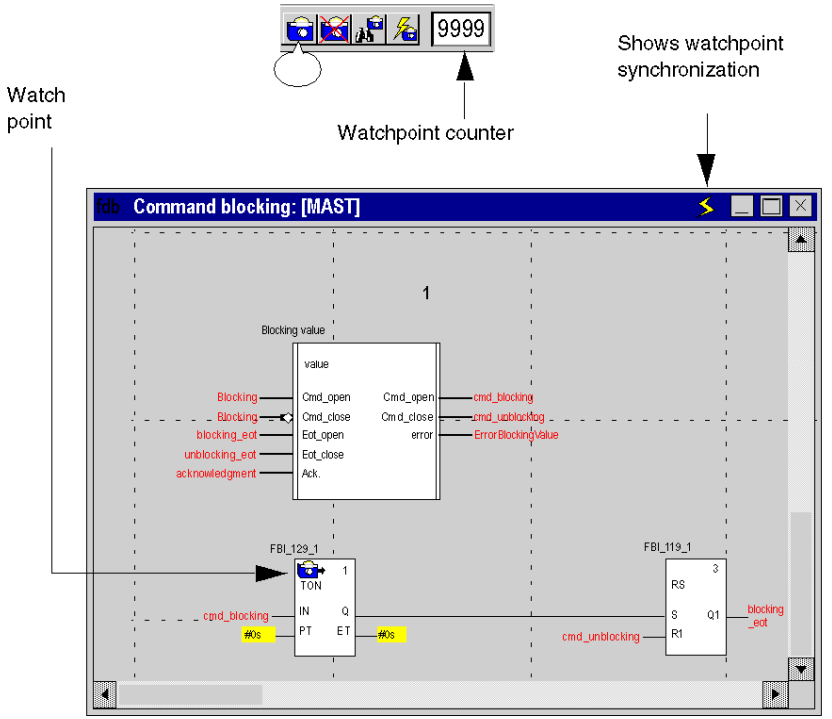
- the watchpoint can be set only in online mode; if the connection is broken, the watchpoint is lost as well,
- a single watchpoint is allowed at a given moment; the watchpoint is exclusive with the breakpoint,
- display of animated variables is effective before execution of the block on which the watchpoint has been set,
- a counter is incremented each time the function block with the watchpoint is executed, up to a maximum value of 9999, then reset to 0,
- modification of a section is not authorized if a watchpoint is set.

## How to Insert a Watchpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the section, select the block desired. |
| 2 | Set the watchpoint in one of the following ways:<br>● by selecting the following command from the menu: **Debug->Set Watchpoint**,<br>● by selecting the following command from the contextual menu: **Set Watchpoint**,<br>● or select the  button in the toolbar. |

Example:




**NOTE:** Inserting a new watchpoint automatically clears the old one.

## How to Locate an Existing Watchpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Either:<br>● by selecting the following command from the menu:<br>**Debug->Show Watchpoint**.<br>● or select the  button in the toolbar. |
| 2 | The part of the language editor containing the watchpoint is displayed. |

## How to Synchronize with an Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the animation table desired. |
| 2 | either you can:<br>● select the following command from the menu:<br>**Debug->Synchronize Animation Table**.<br>● or select the  button in the toolbar. |

## How to Delete a Watchpoint

Either you can:

- select the **Debug->Clear Watchpoint** command from the menu,
- or select the  button in the toolbar.

# Section 38.6
# Debugging User Function Block (DFB) Instances

## Subject of this Section

This section introduces the different tools available and explains how to implement them to debug the DFB instances of the program. These tools are:

- the breakpoint,
- step by step execution,
- the watchpoint.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Inserting a breakpoint in a DFB instance (user function block) | 1451 |
| Step by step execution mode in a DFB instance (user function block) | 1453 |
| Inserting a watchpoint in a user function block (DFB) | 1454 |

# Inserting a breakpoint in a DFB instance (user function block)

## At a Glance

A breakpoint can be used to **stop the execution** of the task which contains the DFB instance in which it has been set.

During debugging it can be used:

- to examine the behavior of the code,
- to view the value of the variables and the input/output parameters of the instance.

There is a single breakpoint at a given point in the project. This is not saved and is lost on disconnection from the PLC.

It is implemented in online mode regardless of whether the PLC is in **Run** or in **Stop**.

## Elements Used for Setting a Breakpoint

The program element used to set a breakpoint in a DFB section depends on the language in which the section has been written:

- the **rung and the nested DFB call** for Ladder Diagram language *(see page 1423)*,
- **the instruction** for textual languages *(see page 1432)*,
- the **Function Block** for Function Block Diagram *(see page 1441)* language.

## How to Insert a Breakpoint

Access the DFB instance section *(see page 1419)* from which you wish to insert a breakpoint.

Depending on the language used in the DFB section, proceed in the following manner:

- for Ladder Diagram language *(see page 1424)*,
- for textual languages *(see page 1433)*,
- for Function Block Diagram language *(see page 1433)*.

If **several instances of a DFB** are used in the application, the breakpoint is valid for all calls as the DFB code is shared by all the instances.

The task corresponding to the first call (order of execution) is stopped when the breakpoint in reached.

By relaunching execution using the **Go** button  the task corresponding to the next call is stopped and so on.

**NOTE:** Inserting a new breakpoint automatically clears the old one.

## How to Locate an Existing Breakpoint

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the following command from the menu:<br>**Debug->Show Breakpoint**. |
| 2 | The part of the language editor containing the breakpoint is displayed. |

## How to Delete a Breakpoint

Either you can:

- select the following command from the menu:
  **Debug->Clear Breakpoint**,
- or select the  button in the debug toolbar.

Erasing a breakpoint does not restart the task. To do this, you have to press  .

# Step by step execution mode in a DFB instance (user function block)

## At a Glance

For a DFB section programmed in:

- Ladder Diagram language *(see page 1425)*,
- textual languages *(see page 1434)*,
- Function Block Diagram language *(see page 1444)*.

## Step Into, Step Out and Step Over

Step by step functions are used in the same way for a section belonging to a DFB as for a conventional section.

**NOTE:** a DFB section may contain a sub-routine (SR).

For a DFB section programmed in:

- Ladder Diagram language *(see page 1426)*,
- textual languages *(see page 1435)*,
- Function Block Diagram language *(see page 1445)*.

## DFB with Several Sections

A DFB may be made up of several sections.

The **Step Out** command can be used to move onto the instance following the DFB, and when applied to the last section allows you to exit the instance.

When you want to quit a DFB instance perform as many **Step Out** commands as there are sections. Program execution will go to the first element after the DFB instance call.

## Rule

In step by step mode the task manager will not detect a **watchdog overflow** for the task containing the DFB instance. It will, however, detect infinite loops.

# Inserting a watchpoint in a user function block (DFB)

## At a Glance

Access the DFB instance section *(see page 1419)*from which you wish to insert a watchpoint.

For a DFB section programmed in:

- ladder language *(see page 1428)*,
- textual languages *(see page 1437)*,
- Function Block Diagram language *(see page 1447)*.

# Section 38.7
# Debugging in Sequential Function Chart Language (SFC)

## Subject of this Section

This section introduces the different tools available and explains how to implement them to debug the program. These tools are:

- the breakpoint,
- step by step execution,
- the watchpoint.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Inserting a breakpoint in Sequential Function Chart language (SFC) | 1456 |
| Step by step execution mode in Sequential Function Chart language (SFC) | 1459 |
| Inserting a watchpoint in Sequential Function Chart language | 1463 |

## Inserting a breakpoint in Sequential Function Chart language (SFC)

### At a Glance

In contrast to other languages, a breakpoint in SFC **does not stop execution** of the task corresponding to the point at which it has been set, but **freezes the chart and automatically activates mode "disable transitions".**.

For **this** SFC section (section of the breakpoint), the PLC remains in "disable transitions" mode until you quit this mode using the **Animation panel** tool.

In "disable transitions" mode, the background color of the editor is blue.

During debugging it can be used:

- to examine the behavior of the code,
- to view the value of variables.

In contrast to other languages, several breakpoints can be set at a given point in an SFC section. They are not saved and are lost on disconnection from the PLC.

They are implemented in online mode, regardless of whether the PLC is in **Run** or in **Stop**.

**NOTE:** It is not possible to set a breakpoint in an event task.

**⚠ WARNING**

**UNEXPECTED APPLICATION BEHAVIOR - SFC BREAKPOINT**

When using breakpoint in SFC, ensure that the task and its associated inputs/outputs create an acceptable application processes behavior.

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

**⚠ WARNING**

**UNEXPECTED APPLICATION BEHAVIOR**

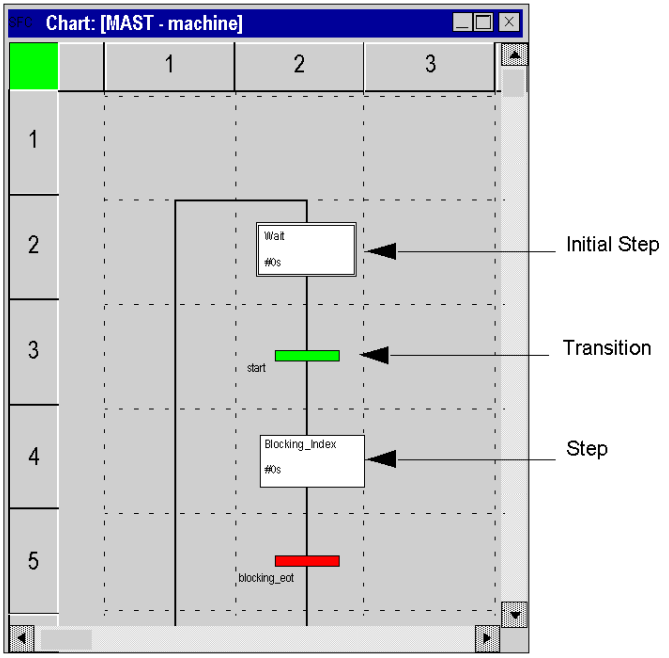
Power off the operating part of the system when using breakpoint in SFC.

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

## Element Used for Setting a Breakpoint

From an **SFC section** the program element used to set a breakpoint is the **step**.

In other words, the following chart section:




From a **section which contains the processing of a step or a transition** the program element used to set a breakpoint corresponds to the language used, i.e.:

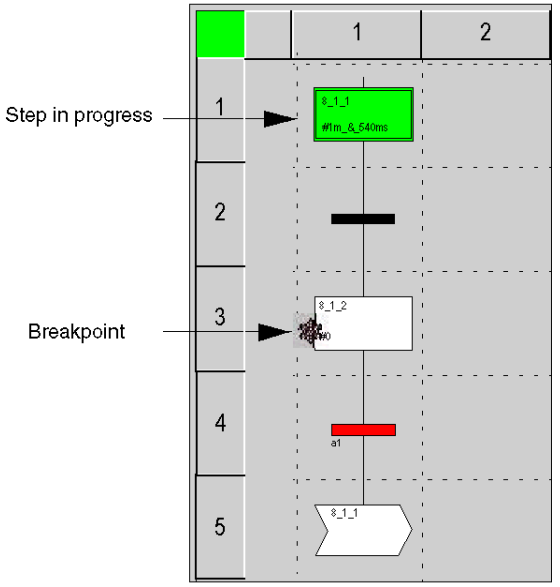
- Ladder Diagram language *(see page 1423)*,
- Structured Text or Instruction List language *(see page 1432)*,
- Function Block Diagram language *(see page 1441)*.

## How to Insert a Breakpoint

From an **SFC section**:

| Step | Action |
|---|---|
| 1 | Select the desired step(s). |
| 2 | Set the breakpoint(s) by right-clicking the desired element and selecting the following command:<br>**Animation → Insert/Remove Breakpoint**. |

Example:




From a **section corresponding to an action or a section corresponding to a transition** the procedures correspond to the language used, i.e.:

- Ladder Diagram language *(see page 1424)*,
- Structured Text or Instruction List language *(see page 1433)*,
- Function Block Diagram language *(see page 1442)*.

## How to Delete a Breakpoint from an SFC section

Select the breakpoint to be deleted, then select the **Debug ->Clear Breakpoint** command from the menu.

**NOTE:** When you initiate a **StepOver** or **StepOut** in a chart, invisible breakpoints are created by SFC editor in the same time than the breakpoint created. In order to remove all these breakpoints use the function **ClearAllBreakpoints**, otherwise the chart execution will stop via **DisableTransition** check button.

# Step by step execution mode in Sequential Function Chart language (SFC)

## At a Glance

In SFC language you have 2 debugging levels:

- from an **SFC section (chart)**,
- from **a section implementing an action or a transition**, where this section is written in one of the following languages (Ladder, Structured Text, Function Block Diagram, Instruction List).

## Sections Implementing Actions or Transitions

In this type of section, debugging behaves in practically the same way as debugging for a conventional section using one of the following languages:

- Ladder Diagram language *(see page 1425)*,
- textual languages *(see page 1434)*,
- Function Block Diagram language *(see page 1444)*.

The specific characteristics of this type of debug are:

- it is impossible to enter a section of an action or a transition in step by step mode when coming from another section, unless this section contains a breakpoint,
- when you are at the end of a section of an action or a transition it is impossible to exit it in step by step mode by using the commands **Step Into, Step Over, Step Out**, **Go** is the only command available.

## SFC Sections (Chart)

Step by step mode consists in executing the application program **step by step**.

**NOTE:** When program execution is over a step, it will not yet have been executed. It will be executed after a command is pressed.

This mode is launched by a **breakpoint** which will have been set in advance (on a step). It is used to examine the behavior of the code, and the value of the variables.

It is implemented in online mode. The section executed in step by step mode in contrast to sections in languages other than SFC does not stop the task, but **freezes the chart**.

The debugging of an SFC section (chart) is checked using the Animation Panel (**Services->Animation Panel**). Each SFC section (chart) has its own tool.

| ⚠ WARNING |
| --- |
| **UNEXPECTED EQUIPMENT OPERATION**<br>The SFC Animation Panel is provided for experienced users only. Changes made via the SFC Animation Panel must be well understood by the user.<br>**Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

## Animation Panel Tool

To launch the tool, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Open the editor for the SFC section (chart) that you wish to debug. |
| 2 | Select the following command from the menu:<br>**Services->Animation Panel->Show Animation Panel**<br>**Result**: The tool is displayed. |

Animation Panel tool:



Description of different commands:

| SFC section name | |
|---|---|
| **Parameters zone** | |
| **Initialize Chart** | For an SFC section which is currently being executed, do the following in order:<br>1) check this box to deactivate the active step(s),<br>2) uncheck this box to activate the initial steps to start the section. |
| **Disable Time Check** | Check this box to disable checking of step execution times. |
| **Disable Transitions** | Check this box to make it possible to freeze the chart in any transition state. This box must be checked to be able to use the following buttons: **Step Trans Dependent**, **Step Over** and **Step Out** |
| **Disable Actions** | Check this box to stop processing of steps. |
| **Operation zone** | |
| **Clear Chart** | Deactivates the active step(s). |
| **Reset Time Errors** | Resets information on step activity time overruns. |
| **Step Unconditional** | Move onto the next step without acknowledgment of the transition condition. Doesn't work if you have a Delay Time (SFCSTEP_TIME.delay) set in the properties of the STEP. |
| **Step Trans Dependent** | Moves to the successor step if the transition condition and the step delay time (SFCSTEP_TIME.delay) are fulfilled. If the transition condition is not right, the chart is unfreezed and waits for the right transition's condition. With the activation of the successor step the chart returns to the freeze state. |
| **Step Over** | This command moves to the successor step if the transition condition and the step delay time (SFCSTEP_TIME.delay) are fulfilled. If the transition condition is not right, the chart is unfreezed and waits for the right transition's condition in order to move onto the next step or steps (in the event of divergence). With the activation of the successor step the chart returns to the freeze state. |
| **Step Out** | Only available if the current element is a step belonging to a macro-step section. This command unfreezes the chart (master and macro belong logically to the same chart) in order to work normally over the end of the macro. The process waits for the valid transition condition(s) of the chart in order to move onto the successor step or steps (in the event of divergence). With the activation of the (first) successor step the chart returns to the freeze state. |
| **Set Break On Selection** | Sets a breakpoint for all selected steps. |
| **Set Pre Pos On Selection** | Sets the pre-positioning markers on the steps and macro-steps of the SFC section. |
| **Set Pre-positioned** | Activates all the steps and macro-steps pre-positioned with the **Set Pre Pos On Selection** command. |

| SFC section name | |
|---|---|
| **Select active steps** | Selects the active steps. |
| **Set Selected Steps** | Activates the selected steps and macro-steps. |
| **Reset Selected Steps** | Deactivates the selected steps and macro-steps.<br>For Multi-Token operating mode only (**Project Settings** → **Allow multiple token**). |
| **Clear All Breakpoints** | Deactivates all the breakpoints. |
| **Clear All Pre-Positions** | Deactivates all the steps and macro-steps pre-positioned. |

**NOTE:** The **Step Into** command is not available when debugging an SFC section (chart).

## SFC save and restore

The `SFC_RESTORE` function block can be used to restart all SFC charts of an application with a given set of active steps at a state, saved before a CPU failure.

In a recovery situation it is possible to set a group of steps that represent a given state in the process and to continue at that point.

(See SFC_RESTORE: SFC Save and Restore *(see Unity Pro, System, Block Library)* in the *System Library*).

# Inserting a watchpoint in Sequential Function Chart language

## At a Glance

SFC sections do not support watchpoints.

However, a **watchpoint** can be set on an element in the **section that contains processing for a step or transition**; in this case, the procedures correspond to those described for the language used:

- ladder language *(see page 1428)*,
- textual languages *(see page 1437)*,
- Function Block Diagram language *(see page 1447)*.

# Section 38.8
# Information on Debugging

## Subject of this Section

This section describes how to access additional information during debugging.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Tracing task execution | 1465 |
| Task states during debugging | 1467 |
| Disconnection during the debug phase | 1468 |
| Operation of the alarm relay in Debug mode | 1469 |

# Tracing task execution

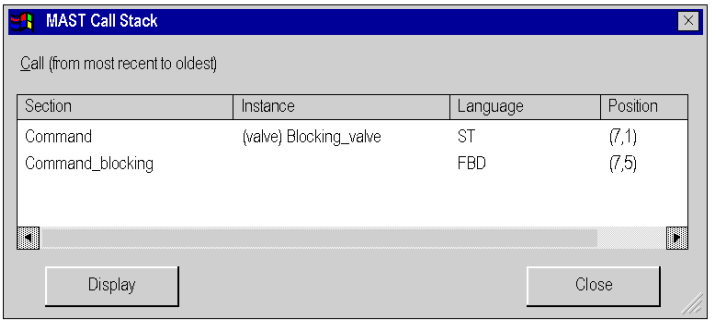
## At a Glance

Tracing task execution consists of knowing at a given moment (breakpoint reached, step by step mode in progress) what the task execution path is, that is to say, what **subroutines (SR)**, **user function blocks (DFB)** have been called and what the nesting level is.

For this purpose, a tool is available for your use: the **LIFO stack** (Last In First Out) which memorizes all nestings so that you can monitor task execution.

## How to Access a Program Element from the Stack

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | From a breakpoint or a step-by-step mode in progress either by:<br>● selecting the following command from the menu: **Debug->Call Stack**,<br>● or by pressing the  button in the debug toolbar.<br>Example: **Command_blocking** section having called a **Blocking_valve (valve type)** DFB instance in the **Command** section:<br>  <br>The line number is used for Structured Text and Instruction List editors; for Ladder and FBD editors, this is the number of the rung or the block that is displayed. |

| Step | Action |
|---|---|
| 2 | If you select:<br>● the **Command** section and you click on **Show**, the editor positions itself on the step in progress in the DFB instance; if this is a breakpoint, it is indicated in the left margin by  ,<br>● the **Command_blocking** section and you click on **Show**, the editor is positioned on the DFB instance call that is not the step in progress: it is indicated by  . |

Clicking the **Close** button closes the window representing the stack: the green triangle disappears following:

- the resumption of step by step,
- a new call to display the contents of the stack,
- a restart of the task  .

# Task states during debugging

## At a Glance

When debugging a project, it can be useful to determine the current state of tasks it contains. Apart from the master task (MAST), these tasks can include:

- the FAST task,
- the auxiliary tasks (AUX0, AUX1, AUX2, AUX3).

The task state can be:

- **HALT** task stopped following the PLC's switch to "Halt",
- **RUN** task currently being executed,
- **BKPT** task in step by step mode,
- **STOP** task stopped.

## How to Determine the Project's Task State

To determine the current task state, select the **Debug->Task State** command from the menu.

Example with a single task (MAST) in step by step mode:

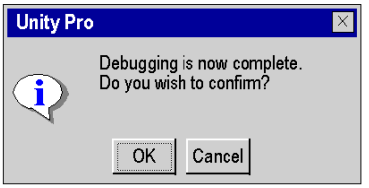


# Disconnection during the debug phase

## Caused by the User

For a variety of reasons, you can decide to switch from **online mode** to **offline mode** even if a breakpoint has been set in a section or if you are in step by step mode.

In this case, you must confirm your choice:




Following confirmation, all application tasks are stopped.

## Caused by a fault

When a breakpoint is set in a task section, or when you are in step by step mode, if a hardware or communication problem is detected, the PLC switches automatically from **online mode** to **offline mode** and the breakpoint is lost without a request for confirmation.

## Operation of the alarm relay in Debug mode

### Description

The alarm relay (or safety output) changes status on every step or breakpoint it meets.

At the end of each cycle, the outputs are refreshed.

| ⚠ CAUTION |
| --- |
| **UNEXPECTED APPLICATION BEHAVIOR - ALARM RELAY**<br>Set the bit %S9 to 1 in order to force outputs to fallback mode to stop alarm relay changes on every step or breakpoint.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

# Section 38.9
# PLC debug screen

## Aim of this sub-section

This sub-section shows the debug screen associated with the PLC processor.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Overview of the CPU debugging screen | 1471 |
| Description of the Task tab: Task checking field | 1473 |
| Description of the Task tab: Operating mode field | 1475 |
| Description of tab: Realtime Clock | 1477 |
| Description of the Tab: Information | 1478 |

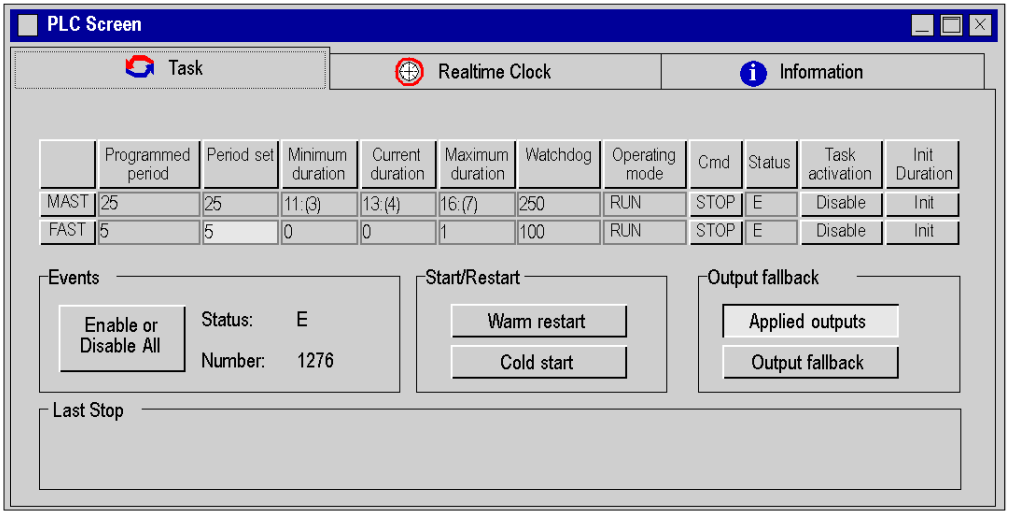
# Overview of the CPU debugging screen

## Role

This screen is composed of 3 tabs:

- **Task**, statuses and execution commands of the program tasks
- **Real-time Clock**, adjusting the real-time clock
- **Information**, information on the project and the PLC processor

PLC debug screens




**NOTE:** In the Minimum duration time , Current duration time, and Maximum duration time fields, the values between parenthesis corespond to the Minimum, Current, and Maximum overhead time *(see Modicon M340 Using Unity Pro, Processors, Racks, and Power Supply Modules, Setup Manual)*.

## How to access the PLC debug screen

The CPU debug screen is only accessible in online mode using the **Tools → PLC Screen** command or by double clicking on the PLC processor in the configuration editor and choosing the **Animation** tab.

If the connection is interrupted, the screen remains displayed but is inactive.

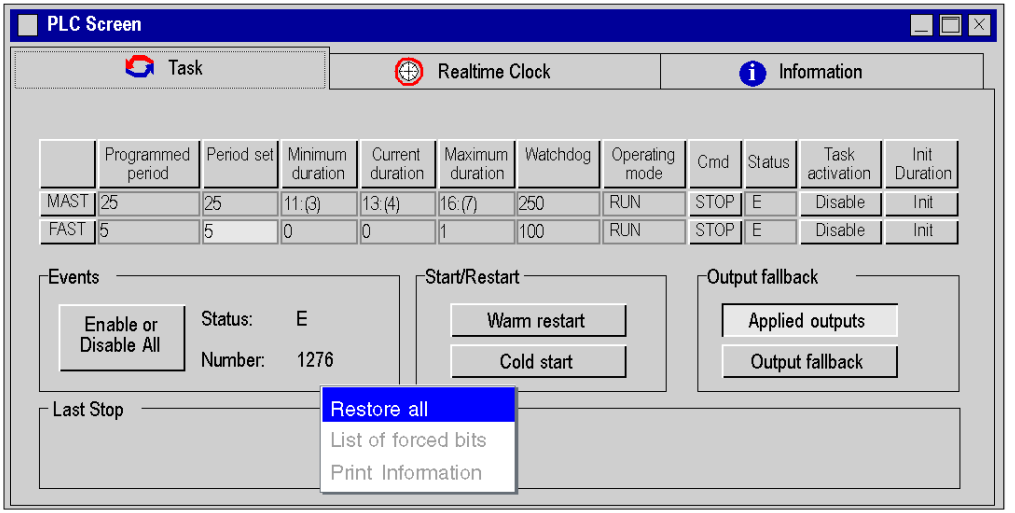
To activate the commands or carry out the adjustments proposed in these tabs, you must have appropriate user rights.

## Contextual menu in PLC screen

The contextual menu in the PLC screen displays three options:

- Restore all *(see page 1473)*, restore task periods configured for all periodic tasks
- List of forced bits, display list of all forced bits
- Print Information, print data from the information tab

PLC debug screens and its associated contextual menu



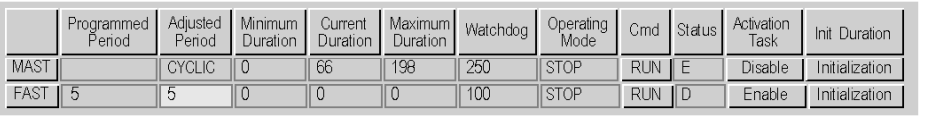

**NOTE:** The **Restore all** option is only available when Task tab is active and **Print Information** when the Information Tab is active. All the options available in the contextual menu are also available in **Services** → **Restore** or **Services** → **Restore all** commands.

# Description of the Task tab: Task checking field

## At a Glance

This field gives you access to the display and settings of the task periods and the various task execution commands.

Task checking field:



| | Programmed Period | Adjusted Period | Minimum Duration | Current Duration | Maximum Duration | Watchdog | Operating Mode | Cmd | Status | Activation Task | Init Duration |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MAST | | CYCLIC | 0 | 66 | 198 | 250 | STOP | RUN | E | Disable | Initialization |
| FAST | 5 | 5 | 0 | 0 | 0 | 100 | STOP | RUN | D | Enable | Initialization |

This field may be extended according to the tasks configured in the project: one line per auxiliary task may be displayed in addition to those in the above screen.

## Description

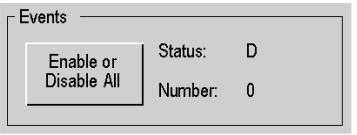
The following table describes the role of each of the screen columns.

| Column | Role |
|---|---|
| Programmed Period | Value of the configured task period (for a periodic task). |
| Adjusted Period | In periodic operation, enables the task period to be adjusted. This setting is lost on a cold start. The value entered in the configuration is then taken into account. Red background color in a field of this column indicates an erroneous returned by the PLC.<br>It is possible to restore task periods configured for the selected task or for all the periodic tasks with the **Services** → **Restore** or **Services** → **Restore all** commands. |
| Minimum Duration | Minimum duration of execution of the task, measured from the last cold restart or reinitialization using the **Init Duration** command. |
| Current Duration | Duration of the last task cycle. |
| Maximum Duration | Maximum duration of execution of the task, measured from the last cold restart or reinitialization using the **Init Duration** command. |
| Watchdog | Value of the task watchdog. This value may not be modified. |
| Operating Mode | Operating mode of the task: RUN, STOP, HALT or BKPT (Breakpoint). |
| Cmd | The **RUN/STOP** buttons associated with the tasks are used to switch to RUN or STOP. |
| Status | • E (enabled): for active task (the system bits %S30 and %S31 are set at 1),<br>• D (disabled): for inactive task (the system bits %S30 and %S31 are set at 0). |

| Column | Role |
|---|---|
| Activation Task | The **Enable/Disable** buttons associated with the tasks are used to enable or disable a task.<br>When a task is **enabled**, the code is scanned and executed. The inputs/outputs are updated.<br>When a task is **disabled**, the code is neither scanned nor executed. The inputs/outputs are still updated. |
| Init Duration | This command is used to reinitialize the values of the maximum and minimum durations with the current value in order to carry out new measurements. |

## Event Processing

The "Events" field is used to check the execution of event processing




The following table describes the role of each of the columns of the screen above.

| Column | Role |
|---|---|
| Enable or disable all | This button is used to enable or disable all the event processing operations. If the events appear when the processing operations are disabled, they are lost. The master task must be in RUN for this command to be authorized. |
| Status | indicates the status of the event processing operations: **E**nabled or **D**isabled. |
| Number | Number of event processing operations executed |

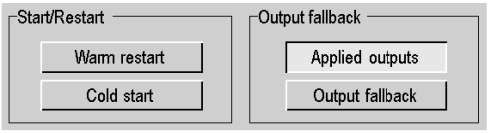
# Description of the Task tab: Operating mode field

## At a Glance

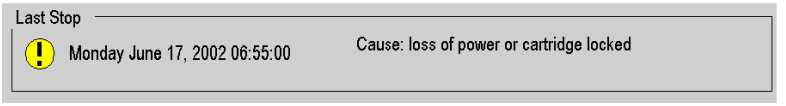
This field gives access to various simulation buttons:

- **Warm restart**,
- **Cold start**,
- **Output fallback**.

Operating Mode field:




This field also indicates the date, time and cause of the last PLC stop.




## Description

Buttons and functions:

| Button | Function |
|---|---|
| Warm restart | Click on this button to perform a warm restart of the PLC.<br>This sets the %S1 bit to 1, which, for example, is used to command a partial initialization program.<br>This does not set %S21 and %S13. |
| Cold start | Click on this button to perform a cold start of the PLC.<br>This initializes the data, the system, and sets the %S0 bit to 1 which (in this case) is used for example to command a specific initialization program.<br>This does not sets %S21 and %S13. |
| Output fallback | It is used to switch the outputs into fallback mode. The values supplied to the outputs are then the values defined in the configuration (fallback value on PLC STOP).<br>When the outputs are in fallback, the **Applied outputs** button becomes active and allows you to exit fallback mode. |
| Applied outputs | Is used to stop the fallback mode and to reapply the values supplied by the program to the outputs. |

## Specific features

**Management of output fallbacks for Premium Atrium, M580 and M340 PLCs**: when a task is set to STOP, the associated outputs automatically switch to the configured mode (fallback or maintain). The outputs associated with the other tasks continue to be updated as normal.

For Premium PLCs, during a warm start, an output in fallback or maintain mode goes to the security mode (state 0) while the CPU is not started. When the CPU is started, if the output falback function is still active, the output will return to the fallback mode or stay at 0 in maintain mode.

For M340 PLCs

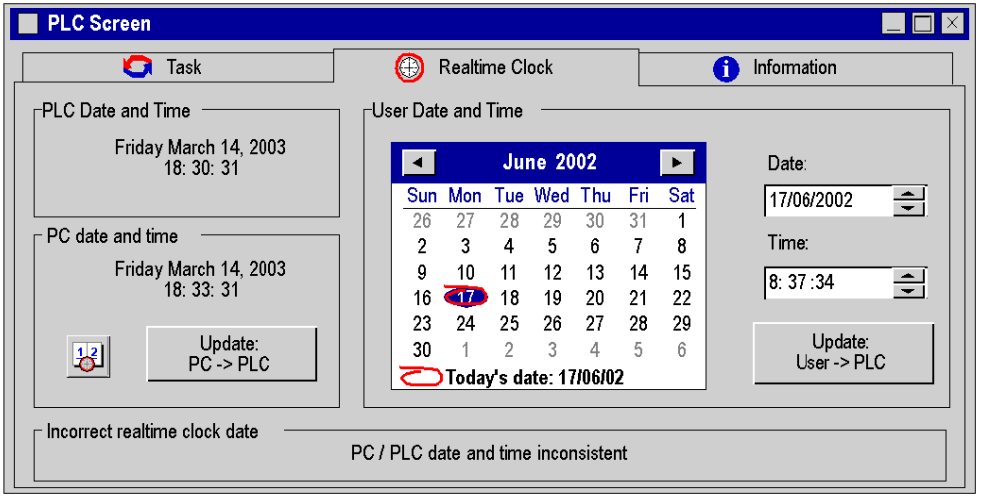
- Whatever the operating mode, an output in fallback or maintain mode do not return to the previous state but stays in security mode (state 0) while the fallback condition is applied.
- The fallback values are only applied if the **Topological** option of **Memory Management** is checked, see in the PLC configuration screen *(see page 1353)*.

**NOTE:** The buttons **Output fallback** and **Applied Outputs** are not available for Quantum PLCs. On Quantum PLCs, the output of the stopped task will continue to be updated (with the last value written), the others are handled normally. Inputs of the stopped task will continue to be scanned, but the data input will be discarded.

# Description of tab: Realtime Clock

## At a Glance

This field gives access to the real-time clock settings.




## Description

Fields and functions:

| Field | Function |
|---|---|
| PLC Date and Time | Displays the date and time of the PLC processor. |
| PC date and time | Displays the date and time of the PC.<br><br>The icon:  is used to access the PC Date and Time setting screen.<br>The **Update PC->PLC** button updates the real-time clock of the PLC in line with the date and time of the PC. |
| User Date and Time | Is used to adjust the date and time using the calendar and the entry fields: **Date** and **Time**.<br>The **Update User->PLC** button updates the real-time clock of the PLC in line with the date and time set by the user. |
| Incorrect realtime clock date | This field displays the cause of the error when a date or time is not accepted. |

# Description of the Tab: Information

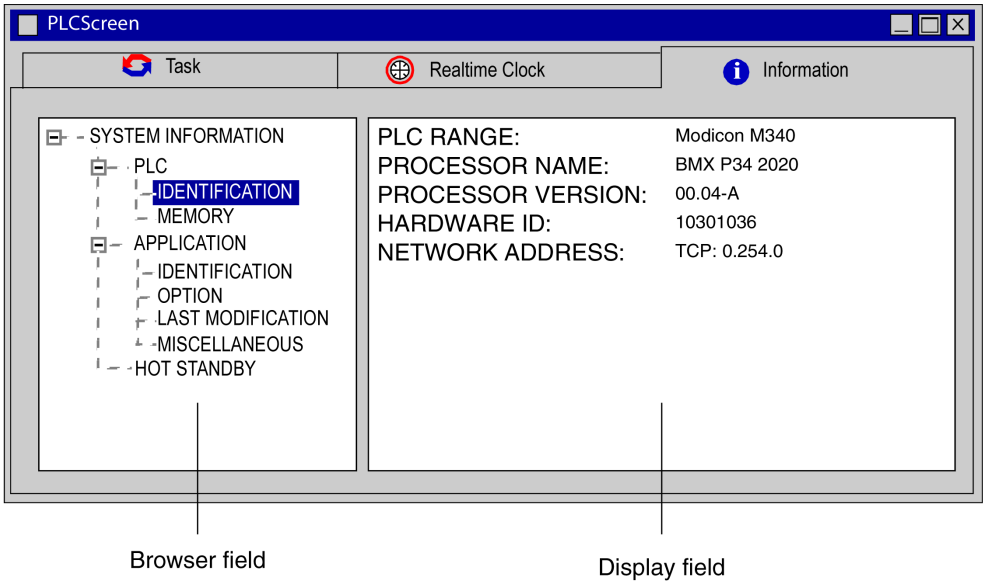
## At a Glance

The **Information** tab is composed of 2 areas:

- the browser field, which is used to choose the subject
- the display field displayed the information concerning the chosen subject is displayed.

The **Services** → **Print Information** command is used to print this information.

Information tab:




**NOTE:**

To view the content of the **Information** tab perform the following actions:

- build the project (**Build Changes** or **Rebuild All Project**)
- connect Unity Pro to the PLC (**PLC** → **Connect**)
- download (**PLC** → **Transfer Project to PLC**)

## PLC IDENTIFICATION Information

This information is used to identify the PLC processor:

| Element | Function |
|---|---|
| PLC RANGE | Type of PLC connected |
| PROCESSOR NAME | Reference of the connected processor |
| PROCESSOR VERSION | Software version of the processor |
| HARDWARE ID | Hardware identifier |
| NETWORK ADDRESS | Connection address |

## PLC MEMORY Information

This information is used to identify the types of memory and their size.

| Element | Function |
|---|---|
| RAM CPU | Size of the internal RAM memory |
| BMX RMSxxx | Part Number of the SD-Card present in the PLC. |
| SD-Card | A memory card is inside the PLC, but Unity Pro cannot read the part number. |
| RAM CARTRIDGE | Type of the RAM cartridge |
| FLASH CARTRIDGE | Type and size of the additional RAM cartridge |

## APPLICATION IDENTIFICATION Information

This information is used to identify the project:

| Element | Function |
|---|---|
| NAME | Project name |
| CREATION PRODUCT | Software version from which the project has been created |
| DATE | Project creation date |
| MODIFICATION PRODUCT | Software version with which the project has been modified |
| DATE | Project modification date |
| VERSION | Project version |
| SIGNATURE | Software signature associated with the project |
| LIBSET VERSION | **Libset** version used for the project |

## APPLICATION OPTION Information

This information describes the content of the project loaded into the PLC.

| Element | Function |
|---|---|
| NEW TERMINAL SUPPORT | Presence or absence of support information for new console (includes "Upload information", "Comments" and "Animation tables"). |
| UPLOAD INFORMATION | Presence or absence of information needed to upload the project to the terminal or to the PLC memory. |
| COMMENTS | Presence or absence of comments on variables in the PLC memory |
| ANIMATION TABLE | Presence or absence of animation tables in the PLC memory. |
| PROTECTION OF SECTIONS | Protection of the project sections |
| DIAGNOSTICS | Depends on PLC diagnostic project setting:<br>● YES if application diagnostic or system diagnostic is checked<br>● NO if application diagnostic and system diagnostic are unchecked |

## APPLICATION LAST MODIFICATION Information

This information describes the general information of the project loaded into the PLC.

| Element | Function |
|---|---|
| MODIFICATION PRODUCT | Software version with which the project has been modified. |
| DATE | Project modification date. |
| PC NAME | Name of the PC[1] |
| APPLICATION | Path and name of the project saved in `*.STU`[2] |
| USER NAME | Display the name of the user profile associated with Unity Pro[1] |
| **1** These elements can be hidden if no information are found<br>**2** This element can be hidden if the project has not been previously saved | |

## APPLICATION MISCELLANEOUS Information

Additional project information

| Element | Function |
|---|---|
| NEW TERMINAL SUPPORT | Update status of support information for new console. |
| FORCED BITS | Display the number of forced bits in the application.<br>A single click on this line starts the animation table containing all the forced bits. This service is only operational if the content of the PLC is identical to the content of the programming terminal and if there are forced bits.<br>See also Animation Table of Forced Bits *(see page 1521)*. |

## HOT STANDBY Information

**Quantum and Premium:** This information displays the Hot Standby status register (%SW61). It is used to monitor the current machine status of the Primary CPU and Standby CPU.
For details about the bits in this register, refer to *Unity Pro System Bits and Words, Reference Manual*.

**Modicon M580:** This information displays the Hot Standby status register part of the Hot standby device DDT T_M_ECPU_HSBY *(see Modicon M580 Hot Standby, System Planning Guide for, Frequently Used Architectures)*.

# Chapter 39
# Viewing and adjusting variables

## Subject of this Chapter

This chapter guides the user through how to implement the tools for viewing and adjusting project variables. These tools are:

- the variable display window,
- the variable display boxes,
- the animation tables.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 39.1 | Introduction to variable animation | 1484 |
| 39.2 | Variable Display Window | 1485 |
| 39.3 | Variable Display Boxes | 1488 |
| 39.4 | Animation Tables | 1491 |
| 39.5 | Instruction Editor (LL984) | 1522 |
| 39.6 | Modifying values: | 1525 |

# Section 39.1
# Introduction to variable animation

## Displaying variables in debug mode

### At a Glance

Variables can be displayed using different language editors (sections) *(see page 1406)*, and additional tools, which are:

- the variable display window,
- the display boxes,
- the animation tables.

### Variables Shown in the Display Window

Certain variables used in a section cannot all be animated directly in the language editor. These are:

- numerical variables in sections that use textual languages (ST, IL),
- numerical variables used in operation and comparison blocks in textual languages (ST, IL),
- structure variables belonging to the DDT family (derived data type),
- parameters and variables belonging to DFB or EFB instances.

These variables are shown in the **display window**, called from the language editors.

The window automatically displays the language editor variables visible onscreen; when the editor runs through the section, the window is updated automatically with the new variables.

### Variables Shown in the Display Boxes

Each box displays the name and value of a single variable.

The boxes can be placed anywhere in the section and can be configured.

### Variables Displayed in the Animation Tables

Certain variables must be continually monitored, regardless of whether they are visible or not in the language editor at a given time.

These variables are displayed in the **animation tables**, called from the language editors.

# Section 39.2
# Variable Display Window

## Introduction to the Variable Output Window

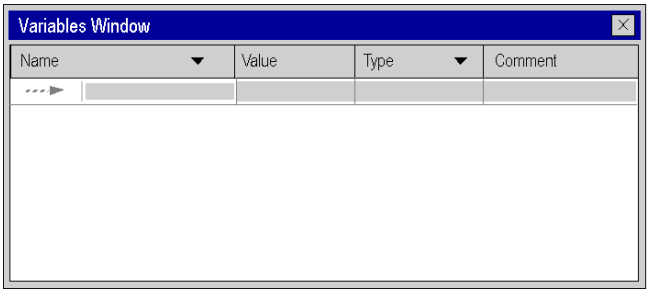
### Opening the Tool

The Output Window is launched from the menu with the

**"Tools->Variables Window"** command.

In online mode, if the language editor is open, the tool automatically contains the variables displayed by the language editor.

Opening in offline mode:




The Output Window includes:

- a line for each variable,
- for each one of the variables, columns describing:
  - their name (symbol or address),
  - their value,
  - their type,
  - a comment.

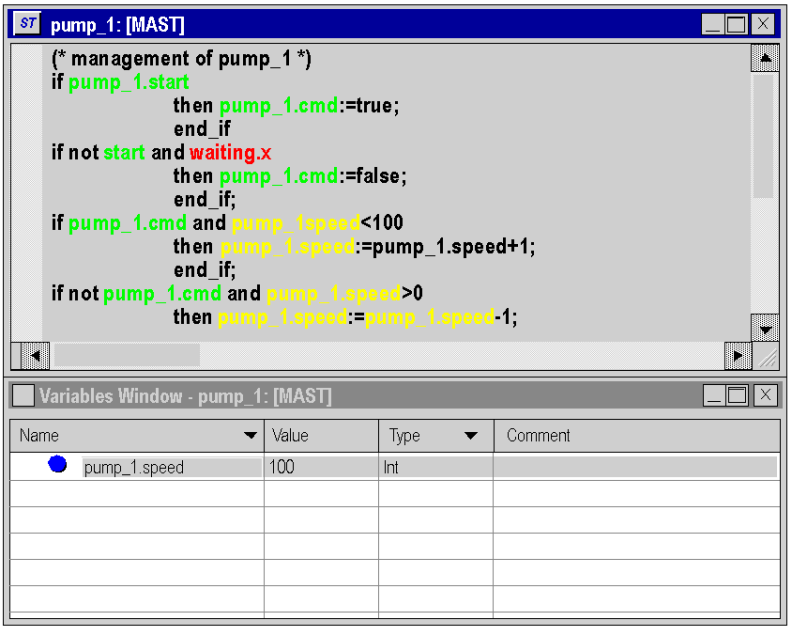
## The Output Window and the Textual Language Editors

A textual language editor (Structured Text, Instruction List) animates Boolean data.

Animation of other types of data takes place in the **Variables Window**; this data includes:

- elementary numerical types,
- derived type (DDT) instance variables, except Boolean types,
- function block (EFB,DFB) instance variables, except Boolean types.

Example with Structured Text editor:




**NOTE:** to display variables of the derived data (DDT) or function block data (EFB\DFB) types, you must expand (+) the name of the corresponding instance.

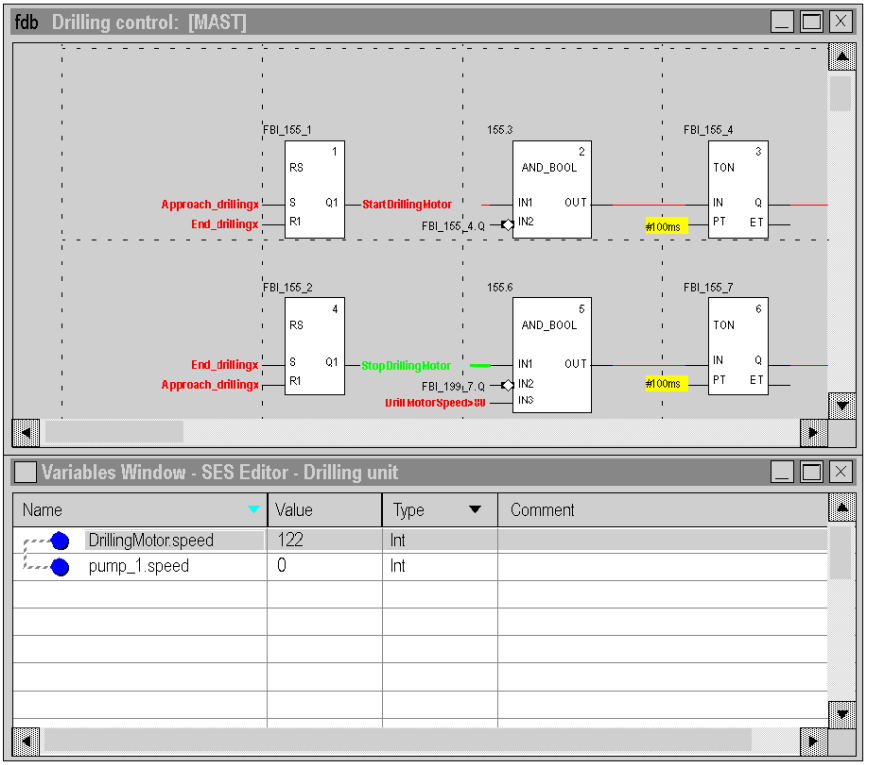
## The Output Window and the Graphical Language Editors

A graphical language editor (LD,FBD) animates Boolean and elementary numerical types of data.

Animation of other types of data can be carried out in the **Output Window**; this data includes:

- derived type (DDT) instance variables, except Boolean types,
- function block (EFB,DFB) instance variables, except Boolean types,
- numerical variables used in the operation or comparison blocks in the Ladder Diagram language editor,
- variables for numerical expressions connected to EFB or DFB in the Ladder or Function Block Diagram language editor.

Example with Function Block Diagram editor:




**NOTE:** to display variables of the derived data (DDT) or function block data (EFB\DFB) types, you must expand (+) the name of the corresponding instance.

# Section 39.3
# Variable Display Boxes

## Introduction to variable display boxes

### At a Glance

Each variable display box displays the name and value of a variable used in the section.

Main characteristics:

- they are called only via sections written in textual languages (Structured Text, Instruction List),
- they contain the variable name and value,
- the boxes can be placed anywhere in the section,
- they can be configured.

The display boxes are implemented in offline or online mode.

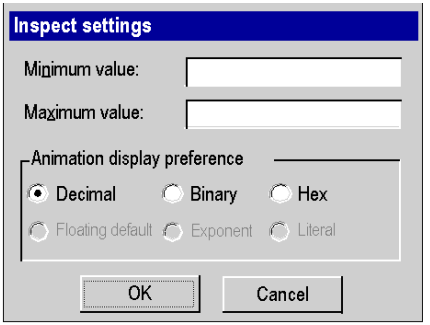
### Creating a Variable Display Box

Carry out the following actions:

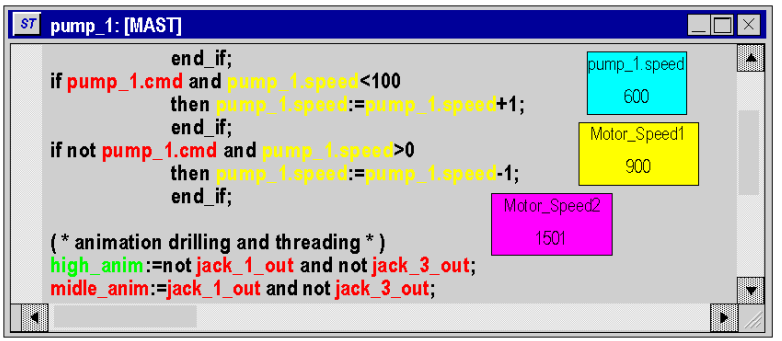
| Step | Action |
|---|---|
| 1 | Select the variable from a section (Structured Text or Instruction List). |
| 2 | From the contextual menu, select the **"New Inspect Window"** command; the box is displayed in the section. |
| 3 | Use the mouse to position the box. |

## Settings

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the display box. |
| 2 | From the contextual menu, launch the **"Settings"** command. |
| 3 | The following dialog box appears:<br>  <br>Enter the minimum and maximum values for the variable:<br>● if the variable value is less than the value entered in **Minimum value**, the display box is shown in **yellow**,<br>● if the variable value falls between the 2 values entered, the display box is shown in **blue**,<br>● if the variable value is greater than the value entered in **Maximum value**, the display box is shown in **pink**.<br>If no value is entered, the box is shown in **blue**. |
| 4 | Select the type of variable display to be used in the display box (Decimal, Binary, Hexadecimal, floating default, exponent, literal) by checking the corresponding box. |

Example:



## Clearing a Variable Display Box

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the display box. |
| 2 | From the contextual menu, launch the **"Delete Inspect"** command. |

# Section 39.4
# Animation Tables

## Subject of this Section

This section guides the user through the implementation of animation tables.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Permanent and Temporary Animation Tables | 1492 |
| Introduction to Animation Tables | 1494 |
| Managing Animation Tables | 1496 |
| Multiple Modifications of Animation Table Variables | 1500 |
| Animation Table Properties | 1501 |
| Basic functions | 1502 |
| Animation of Strings in the Animation Table | 1506 |
| Modification Mode | 1507 |
| Force Mode | 1512 |
| Multiple modification and forcing of Boolean variables | 1518 |
| Animation Table of Forced Bits | 1521 |

# Permanent and Temporary Animation Tables

## Overview

There are 2 kinds of animation tables:
- permanent animation tables
- temporary animation tables

## Permanent Animation Table

A permanent animation table is an integrated part of a project.

It is saved with the project and can be exported.

## Temporary Animation Table

A temporary animation table
- is not saved with a project but deleted when closing a project
  Therefore no temporary animation table is available in the project browser, when opening a project.
- does not modify a project
- does not modify the build state of a project
- does not create upload information (and therefore can not be uploaded)
- can not be exported
- is not included in the printout of a project

## Permanent/Temporary

- Creating an animation table via **Project Browser** → **Animation Table** you can select if a permanent or a temporary animation table should be created. To create a temporary animation table set the **Temporary Table** checkbox in the **Properties** dialog.
- Creating an animation table outside the project browser (e.g. via **Initialize Animation Table** in a language editor or creating an animation table of forced bits) you will always create a temporary animation table.
  See also *Animation Table of Forced Bits, page 1521*.

## Switching Between Permanent and Temporary

Every temporary animation table (except the animation table of forced bits) can be changed to a permanent animation table and vice versa.

There are 2 ways to do this:
- Use the checkbox available in the **Properties** dialog of an animation table (see *Animation Table Properties, page 1501*).
- Use the  button top right at the animation table. Click the button to change from a permanent animation table to a temporary animation table and vice versa.

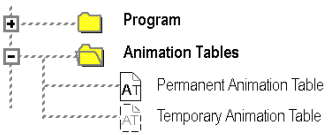
## Context Menu

The context menu that can be opened via **Project Browser → Animation Tables** provides 2 additional entries:

- **Make All Tables Permanent**
  Use this context menu entry to change all temporary animation tables to permanent animation tables.
  As mentioned above, an animation table of forced bits is always a temporary one and can not be changed to a permanent.
- **Purge Temporary Tables**
  Use this context menu entry to delete all temporary animation tables.

## Icons in the Project Browser

In the **Project Browser** the 2 kinds of animation tables are represented by different icons.



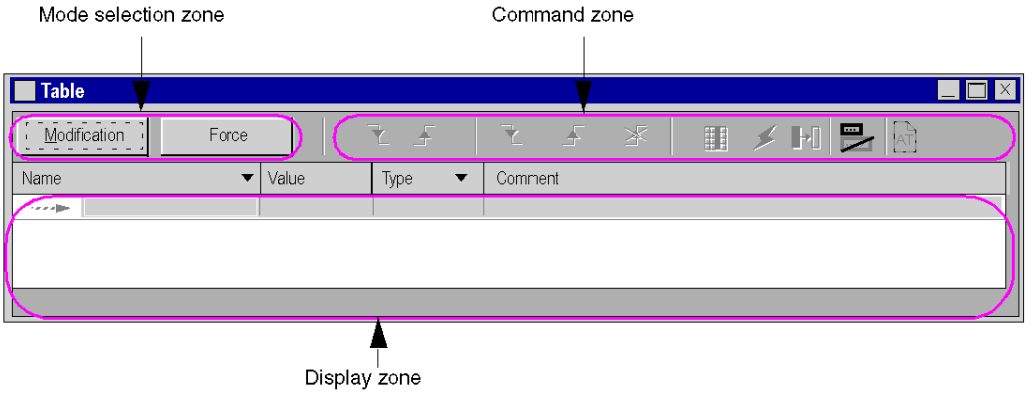
# Introduction to Animation Tables

## Overview

An animation table is divided into 3 areas that include:

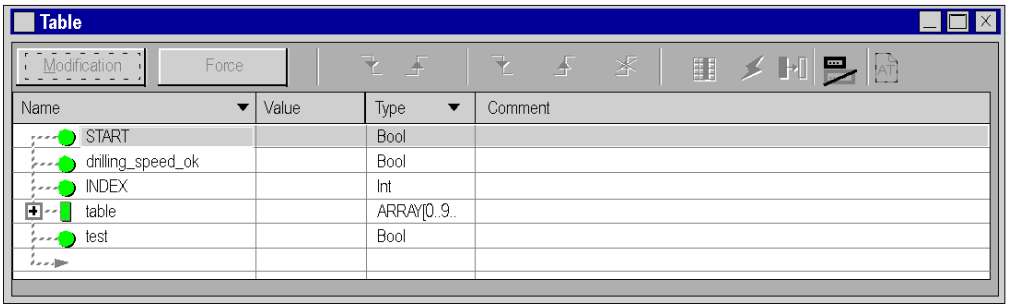
- the **Mode** area,
- the **Command** area,
- the **Display** area,

Animation table:

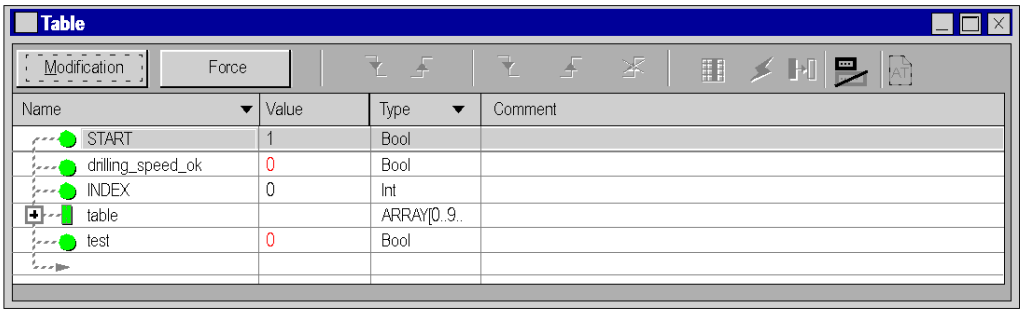



## Online and Offline Mode

In offline mode, an animation table does not display values associated with the variables:



In online mode, an animation table displays the values of the variables.



# Managing Animation Tables

## Creating a New Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the project browser, go to the **Animation Tables** directory. |
| 2 | From the contextual menu; select the **New Animation Table** command; the table is created with a default name. |
| 3 | If you wish, you can rename the table by clicking on its name in the project browser. |

## Opening an Existing Animation Table

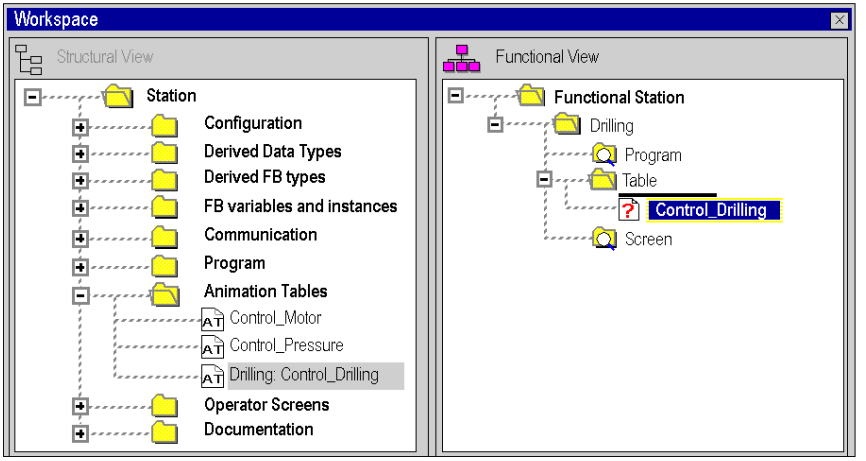
Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the desired table in the **Animation Tables** directory. |
| 2 | Either:<br>● Select the **Open** command from the contextual menu,<br>● or double-click on the name of the table. |

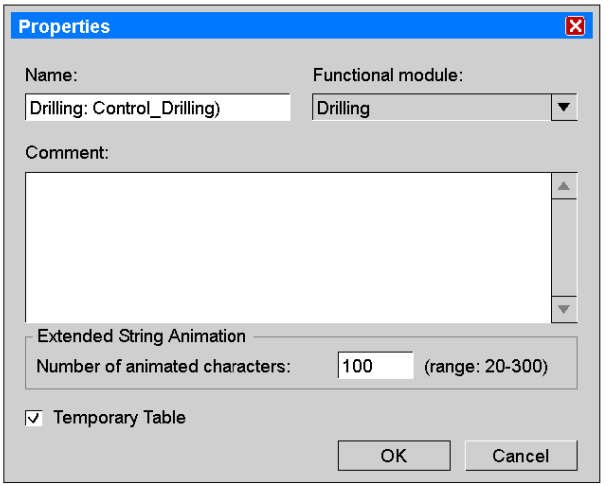
Variables that were displayed previously in the table are re-loaded into the table.

## Attaching an Animation Table to a Functional Module

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Display the **Structural View** and the **Functional View**. |
| 2 | Select the desired table in the **Animation Tables** directory in Structural View. |
| 3 | From the Structural View, use the mouse to **drag\drop** the animation table into the desired functional module in the Functional View. The authorized insertion point of the animation table in the functional module, is represented by a line.<br>Example:<br>  |

Another way to attach an animation table to a functional module is described in the following table:

| Step | Action |
|---|---|
| 1 | Display the **Structural View** . |
| 2 | Select the desired table in the **Animation Tables** directory in Structural View. |
| 3 | Select the **Properties** command from the contextual menu. |
| 4 | Select the desired functional module from the **Functional Module** data selection box.<br>Example:<br>  |
| 5 | Confirm with **OK**. |

## Backing Up an Animation Table

The backup is performed automatically.

## Deleting an Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the desired table in the **Animation Tables** directory. |
| 2 | Select the **Delete** command from the contextual menu. |

## Properties of the Animation Table

To access the properties of the animation table, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Select the desired table in the **Animation Tables** directory or in the editor window. |
| 2 | Either:<br>● Select the **Properties** command from the contextual menu,<br>● or click **Edit** → **Properties** in the menu bar.<br>Refer also to *Animation Table Properties, page 1501*. |

# Multiple Modifications of Animation Table Variables

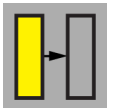
## Entering Multiple Mode

To enter this mode follow this procedure:

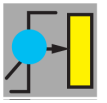
| Step | Action | Icon |
|---|---|---|
| 1 | In the animation table dialog click the **Modification** button: |   |
| 2 | Click the **Enter Multiple Mode** button: |  |

## Current and Set Values

Since Unity Pro V6.0, the user can:

| | |
|---|---|
| Update **Current Values** in the PLC with **Set Values** by clicking: |  |
| Reset all **Set Values** by clicking: |  |
| Save, import and export animation tables with all their **Set Values**. | |

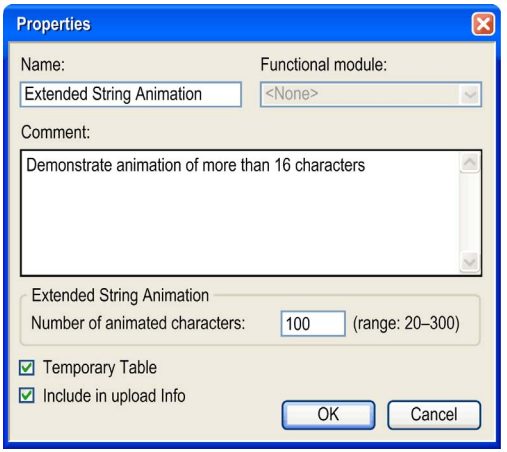
Now, with Unity Pro V8.0, the user can now:

| | |
|---|---|
| Populate the **Set Values** with the **Current Values** by only one click on: |  |

**NOTE:** This new Unity Pro 7.0 feature allows users to manage animation tables as "recipes".

# Animation Table Properties

## Presentation



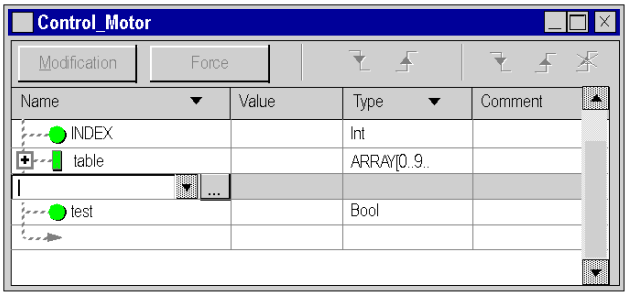

## Animation Table Elements

| Element | Description |
|---|---|
| **Name** | Animation table name |
| **Functional module** | Attached functional module |
| **Comment** | Animation table comment |
| **Number of animated characters** | Number of characters that can be animated in the **Extended Strings** mode (range: 20 ... 300)<br>Refer also to Animation of Strings in the Animation Table *(see page 1506)*. |
| **Temporary Table** | Type of animation table (temporary/permanent)<br>See also Permanent and Temporary Animation Tables *(see page 1492)*. |
| **Include in upload info** | The animation table (including comments) is stored in the *.STU or *.STA application file.<br>This box is checked by default and is:<br>● enabled when **Animation Tables** in the **Project Settings** dialog is **also** checked<br>● **not** enabled when **Animation Tables** in the **Project Settings** dialog is **not** checked |

# Basic functions

## Adding a Data Item to the Animation Table

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Move to an empty line |
| 2 | Either you can:<br>● double-click on the empty line<br>● move to a variable and from the contextual menu, by selecting the **"Insert"** command<br>● type the variable address<br>Example:<br>  |
| 3 | Enter the variable name or select it in the **"Instance selection"** window<br>launched with the  button. |

**NOTE:** Variables previously created in the Data Editor *(see page 325)* can be added to the Animation table by entering the address in the **Name** element of the Animation Table.

## Adding a Data Item From Another Animation Table

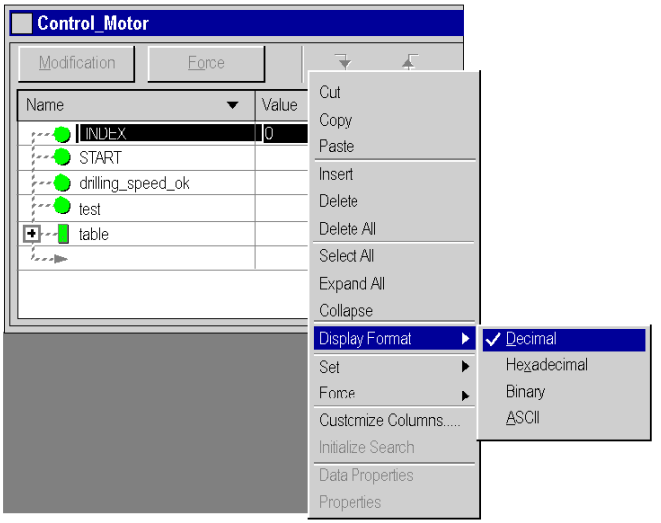
Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the **source animation table**, select the desired data item. |
| 2 | Either:<br>● select the **"Copy"** command from the contextual menu, and in the **destination animation table**, in the desired line, select the **"Paste"** command.<br>● or using the mouse, **drag** the data item and **drop** it in the desired line in the **destination animation table**. |

**NOTE:** You can also carry out the same manipulation for multiple selections. Warning: the drag/drop operation removes the data from the source table and places it in the destination table, whereas Copy/Paste copies the data from the destination table without removing it from the source table.

## Modifying the Display Format

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | In the animation table, select the desired data item. |
| 2 | In the contextual menu, select the **"Display Format"** command and choose the format.<br>Example:<br>  |

**NOTE:** Depending on the display format selected, (Binary, Hexadecimal, ASCII, String, etc.) the value of the data item is preceded by a **prefix** and followed by a **suffix** (except for Decimal format).

## Deleting a Variable

Carry out the following actions:

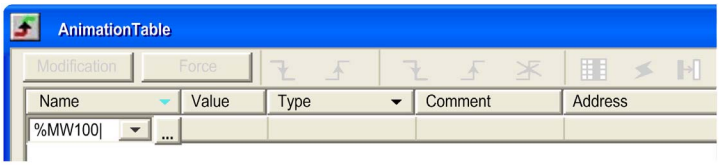
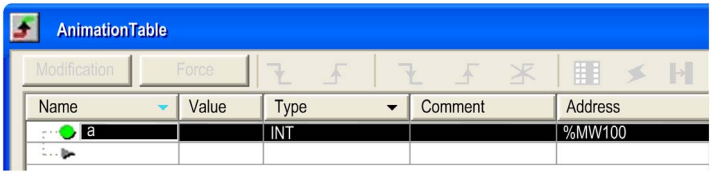
| Step | Action |
|---|---|
| 1 | In the animation table, select the variable(s) desired. |
| 2 | Select the **"Delete"** command from the contextual menu. |

**NOTE:** Once the deletion has been made, the table is reorganized to fill the empty lines.

## Displaying a Variable Name

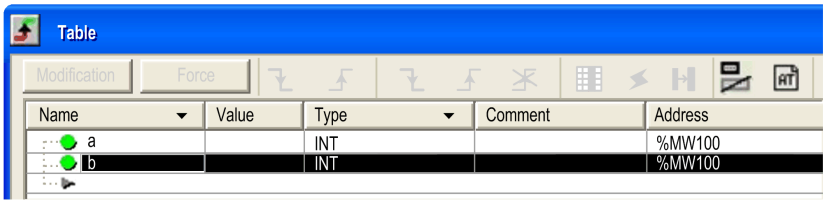
Entering the variable address in the **Name** element displays automatically the variable name.

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Move to an empty line |
| 2 | Either you can:<br>● double-click on the empty line<br>● type the variable address<br>Example:<br>  <br><br>**Result**: Animation table automatically displays a in the **Name** element:<br>  <br> |

**NOTE:** It is possible to have more than one variables with the same address entered in the **Name** element. The Animation Table will create an new entry for each variables.

The graphic below shows two variables at the %MW100 address:



# Animation of Strings in the Animation Table

## Overview

The animation table offers two different modes (**Modification**/**Force**) to animate strings.

Normally only the first 16 characters of a string are animated (modified).

The reason for this restriction is that for animation handling a PLC reserves a buffer.

The size of this buffer is limited and the animation of complete strings can exceed the size of the buffer easily.

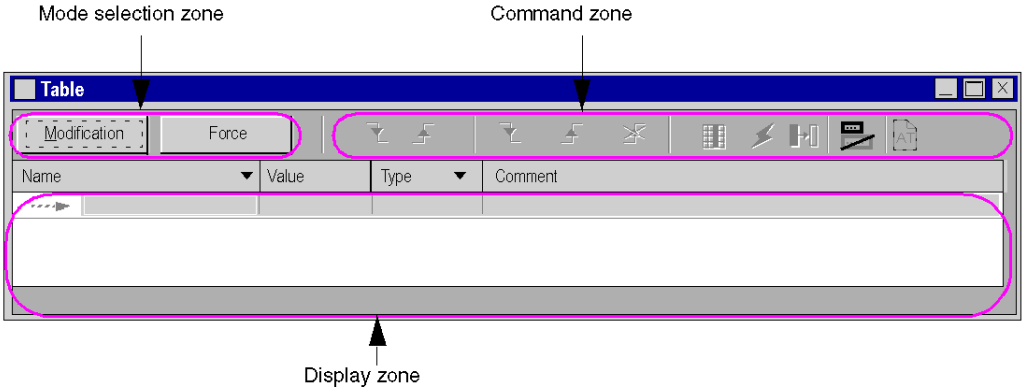
## Extended Strings

The animation table can animate more than 16 characters of a string.

At the right side of the command zone area there is the checkbox  (Extended Strings).

It can be used to activate the mode **extended string animation**.

## Presentation



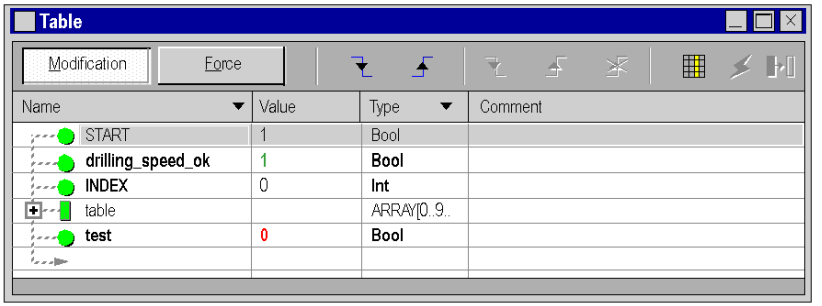

The number of characters to be animated in the mode **extended string animation** can be set in the properties of the animation table (see *Animation Table Properties, page 1501*).

**NOTE:** Using this feature you should keep in mind that the animation of big strings can reduce the number of sections and animation tables which can be animated at the same time.

# Modification Mode

## Implementing Modification Mode

Press the **Modification** button in the Mode area:

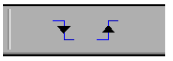



## Modifying an Elementary Variable (EDT)

To modify an **immediate value**, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Double-click in the **"Value"** column on the line corresponding to the variable that you wish to modify. |
| 2 | Use the keyboard to enter the desired value. |
| 3 | Confirm your choice by pressing the **"Enter" key.** |

To modify a **Boolean value**, carry out the following actions:

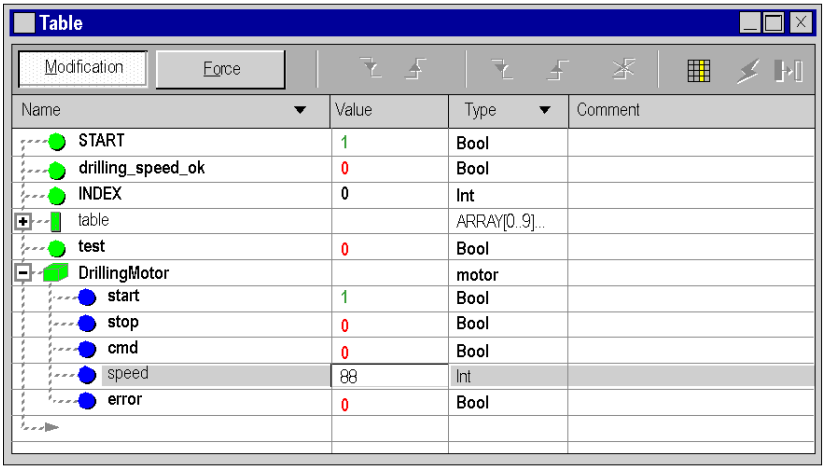
| Step | Action |
|---|---|
| 1 | Use the mouse to select the Boolean value to be modified. |
| 2 | Press one of the buttons  corresponding to the desired value, or from the contextual menu, execute the **"Set to 0" or "Set to 1""** commands.<br>To modify several variables simultaneously, see *Modifying Several Boolean Variables, page 1518* |

## Modifying a Derived Variable (DDT)

Derived variables correspond to structures and to tables.

For a structure, if the elements that compose it are elementary variables, modifications are made element by element see *Modifying an Elementary Variable (EDT), page 1507*.

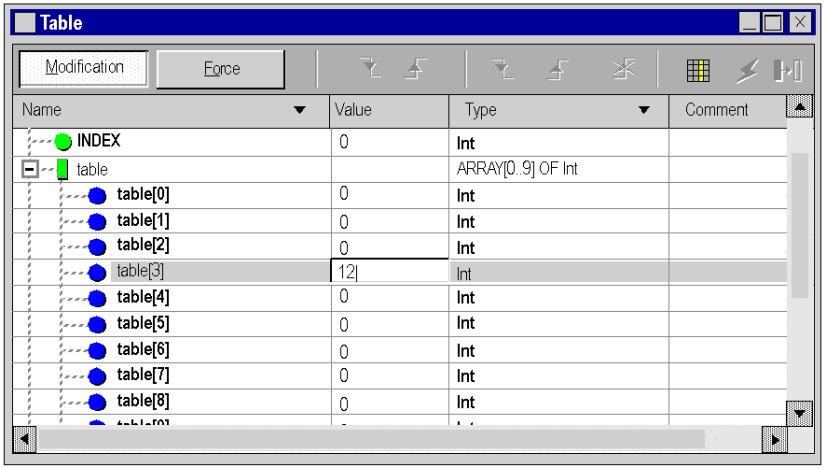
Example:




**NOTE:** if the structures are nested, you must expand them (+) until you reach the elementary variables.

For a table, if the elements that compose it are elementary variables, modifications are made element by element see *Modifying an Elementary Variable (EDT), page 1507*.

Example:



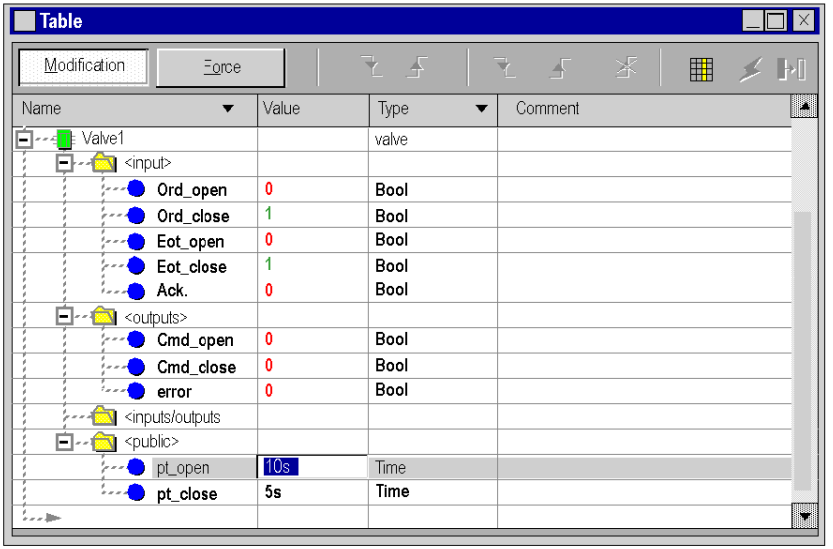

**NOTE:** If the table contains structure-type elements, follow the procedure described above.

## Modifying a Function Block (FB) Data Item

Function block data corresponds to DFBs and EFBs.

If the elements that compose the function block (input/output parameters, public variables) are elementary variables; the modifications are carried out element per element see *Modifying an Elementary Variable (EDT), page 1507*.

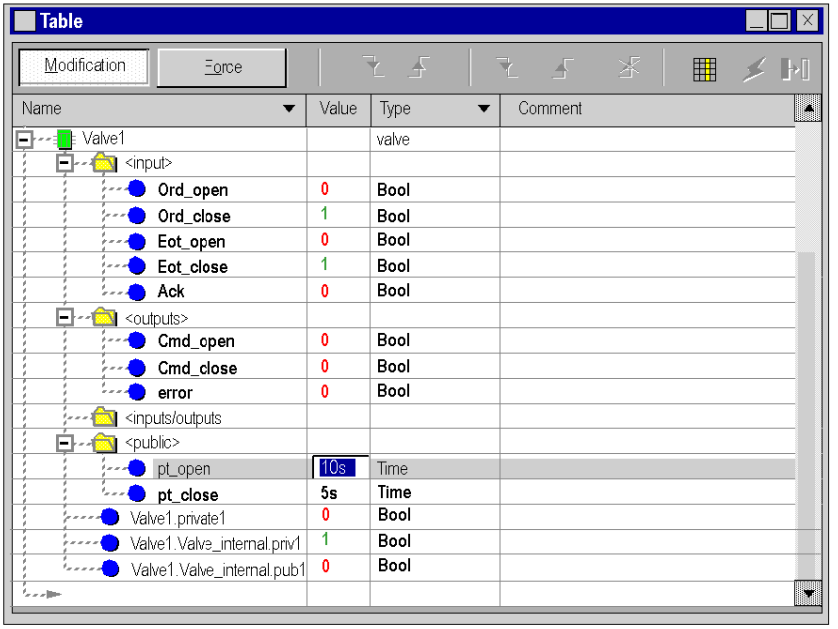
Example of a DFB instance:



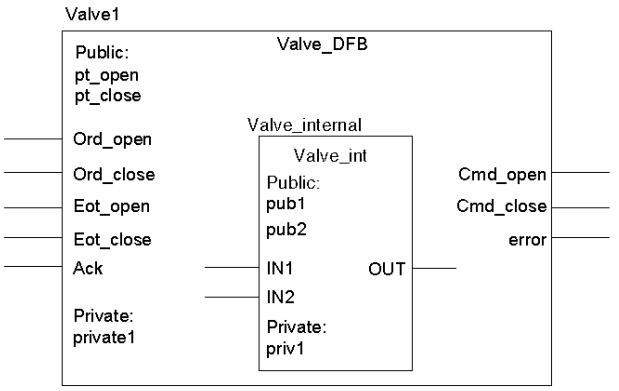

**NOTE:** If the variables are derived (structures, tables), you must expand them (+) until you reach the elementary variables.

Private variables of derived function blocks cannot be accessed by the pre-defined elements. To display or modify private variables you have to enter them variable by variable at the end of the function block tree in the animation table. With the same syntax is is also possible to enter variables that are used in nested derived function blocks.

Example of a DFB instance with private variables:




Example of nested DFB with public and private variables:



# Force Mode

## Preset Force Values in Offline Mode

The **Modification Values** and **Force Values** for variables can be preset in offline mode via animation tables. This information can be transferred to PLC later.

Presetting values can also be used for multiple modification and forcing of Boolean variables *(see page 1518)*.

## Preset Force Values in Project Information

The preset force values of variables are included in the

- download (**Transfer Project to PLC**) information of a project
- upload (**Transfer Project from PLC**) information of a project.

The **Modification Value**/**Force Value** attribute of a variable is part of the animation table information.

## Include in Upload Information

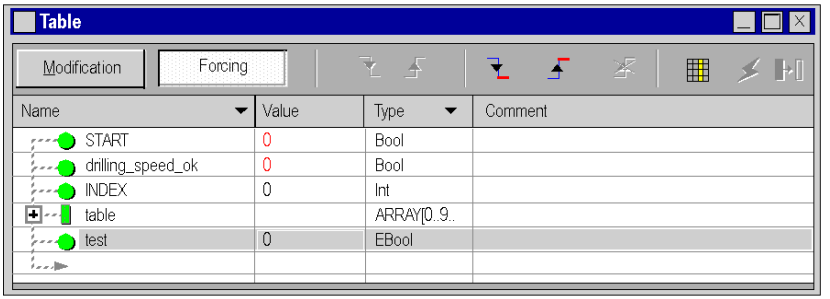
In the properties dialog of an animation table you can activate/deactivate an additional property (**Include in Upload Information**).

As a default this property is activated.

This check box in the property dialog of an animation table is enabled only, if **Animation tables** is activated via **Project Settings** → **General** → **PLC embedded data** → **Upload information**.

## Implementing Forcing Mode

Press the **Force** button in the **Operating Mode** area:



This mode is only available for variables responding to these conditions:
- The variable is an `EBool` type.
- The variable is a located variable.
- The forcing attribute is validated in the variable editor.

**NOTE:**
Analog I/O variables forcing is managed in a different way:
- On analog Device DDT *(see Modicon M340 with Unity Pro, Analog input/output modules, User manual)*, a set of sub fields (for example MOD_ANA_8_1.ANA_CH_IN[0].ANA.FORCE_CMD) allows to force analog I/Os with a predefined value.
- For Modicon M580 and M340 PLCs the analog I/Os variables can be forced in the program.

**NOTE:** The **FORCED I/O** led of the M580 will switch to 1 when forced is active.

**NOTE:** With the animation table, it is not possible to set or reset the bit associated to step S.x.

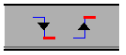
Forcing input and output values in a running controller can have serious consequences to the operation of a machine or process. Only those who understand the implications in the controlling logic, and who understand the consequences of forced I/O on the machine or process, should attempt to use this function.

| ⚠ WARNING |
| --- |
| **UNINTENDED EQUIPMENT OPERATION**<br>You must have prior knowledge of the process and the controlled equipment and the modified behavior in Unity Pro before attempting to force I/O or numeric values to memory locations.<br>**Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

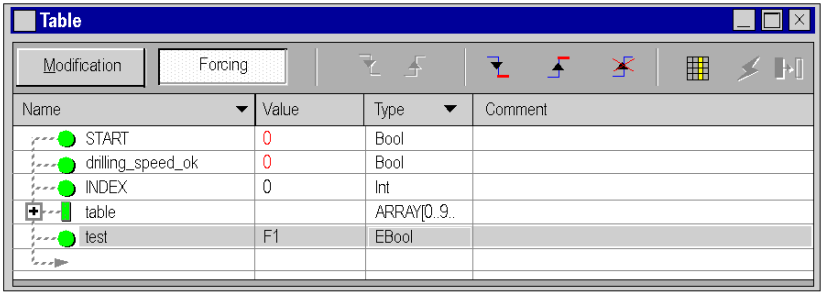
**NOTE:**
Before forcing outputs, understand that forced outputs (%M) exhibit different behavior with Modsoft/NxT/Concept than Unity Pro.
- With Modsoft/NxT/Concept you **cannot** force outputs when the Memory Protect switch of the Quantum CPU is set to the "On" position.
  With Unity Pro you **can** force outputs even when the Memory Protect switch of the Quantum CPU is set to the "On" position.
- With Modsoft/NxT/Concept forced outputs **maintain** their values following a cold start.
  With Unity Pro forced outputs **lose** their disabled state following a cold start.

## Forcing a Variable

To force a localized Boolean variable, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Use the mouse to select the Boolean variable. |
| 2 | Press one of the buttons  corresponding to the desired value, or from the contextual menu, execute the **Force to 0** or **Force to 1** commands.<br>To force several variables simultaneously, see Forcing several located variables *(see page 1519)*. |

When a variable is forced, the letter **F** is displayed in front of the value:




Forcing input and output values in a running controller can have serious consequences to the operation of a machine or process. Only those who understand the implications in the controlling logic, and who understand the consequences of forced I/O on the machine or process, should attempt to use this function.

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

You must have prior knowledge of the process and the controlled equipment and the modified behavior in Unity Pro before attempting to force I/O or numeric values to memory locations.

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

**NOTE:**
Before forcing outputs, understand that forced outputs (%M) exhibit different behavior with Modsoft/NxT/Concept than Unity Pro.

- With Modsoft/NxT/Concept you **cannot** force outputs when the Memory Protect switch of the Quantum CPU is set to the "On" position.
  With Unity Pro you **can** force outputs even when the Memory Protect switch of the Quantum CPU is set to the "On" position.
- With Modsoft/NxT/Concept forced outputs **maintain** their values following a cold start.
  With Unity Pro forced outputs **lose** their disabled state following a cold start.

## Unforcing a Variable

To unforce a located Boolean variable, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Use the mouse to select the Boolean variable. |
| 2 | Press the  . button or execute the **Unforce** command from the contextual menu.<br>To unforce several variables simultaneously, see Unforcing several located variables *(see page 1519)* |

When a variable is unforced, the letter **F** no longer appears in front of the value.

## Creating an Offline Forcing Animation Table

To create a new animation table of forced bits from scratch, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Create a new (permanent) animation table via the **Project Browser**. |
| 2 | Enter the memory bits you want to force either as single bits (like %M1) or as a bit array (like M1:100). |
| 3 | Press the **Force** button. |
| 4 | Press the **Swaps to multiple mode**  button. |
| 5 | Enter forcing values **0** or **1** in the **Force value** column. |

To create a new animation table of existing forced bits in the PLC, carry out the following actions:

| Step | Action |
|---|---|
| 1 | There are 3 ways to open the **Forced bits** animation table:<br>● In the status bar click the red **F**.<br>● In the **Project Browser** right-click **Animations Tables** and select **Open Forced Bits Table** from the context menu.<br>● Via **Tools** → **PLC Screen** open the **PLCScreen** dialog. In the **Information** tab under **APPLICATION** → **MISCELLANEOUS** click the **FORCED BITS** line. |
| 2 | Since this animation table is temporary only, copy and paste the animation table via context menu of the **Project Browser**. |
| 3 | Make the copied animation table permanent (using the **Temporary table** button  ). |
| 4 | In the **copied Forced bits** animation table press the **Force** button. |
| 5 | Press the **Swaps to multiple mode**  button. |
| 6 | Press the **Populates multiple values using existing forced bits**  button.<br>**Result**: The **Force value** column is populated. |

**Build/Build changes** and **Save** your application.

By building, the values entered in the **Force value** column will be included in the upload information which is downloaded into the PLC.

## Forcing by Use of an Offline Forcing Animation Table

To force values before starting the PLC, carry out the following actions:

| Step | Action |
|---|---|
| 1 | Open an existing application (**.STU*, **.STA* or an uploaded application). |
| 2 | Connect to the PLC. |
| 3 | Open the manually created/copied **Forced bits** animation table (see *Creating an Offline Forcing Animation Table* above). |
| 4 | Press the **Force** button to enter forcing mode. |
| 5 | Press the **Swaps to multiple mode**  button. |

| Step | Action |
|---|---|
| 6 | Press the **Execute multiple**  button to force the bits from the preset **Force value** column. |
| 7 | Now you can start your application. |

Forcing input and output values in a running controller can have serious consequences to the operation of a machine or process. Only those who understand the implications in the controlling logic, and who understand the consequences of forced I/O on the machine or process, should attempt to use this function.

## ⚠ WARNING

**UNINTENDED EQUIPMENT OPERATION**

You must have prior knowledge of the process and the controlled equipment and the modified behavior in Unity Pro before attempting to force I/O or numeric values to memory locations.

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

**NOTE:**
Before forcing outputs, understand that forced outputs (%M) exhibit different behavior with Modsoft/NxT/Concept than Unity Pro.

- With Modsoft/NxT/Concept you **cannot** force outputs when the Memory Protect switch of the Quantum CPU is set to the "On" position.
  With Unity Pro you **can** force outputs even when the Memory Protect switch of the Quantum CPU is set to the "On" position.
- With Modsoft/NxT/Concept forced outputs **maintain** their values following a cold start.
  With Unity Pro forced outputs **lose** their disabled state following a cold start.

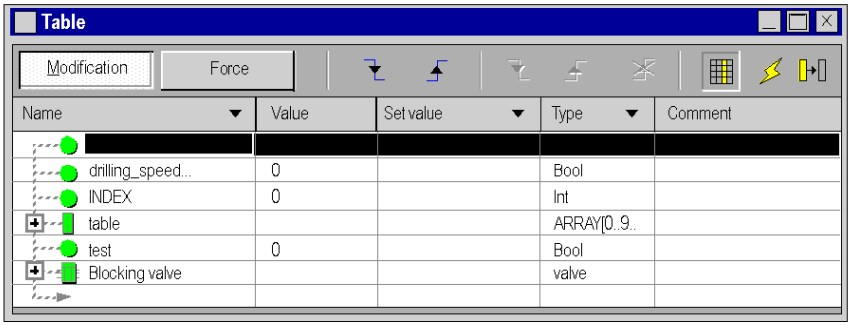
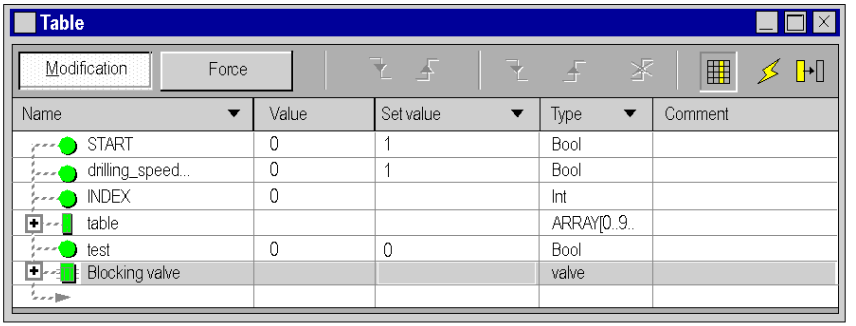
# Multiple modification and forcing of Boolean variables

## At a Glance

The reason for modifying or forcing a group of several Boolean variables (located) is to enable the PLC to acknowledge these changes in **the same cycle**.

## Modifying Several Boolean Variables

Carry out the following actions:

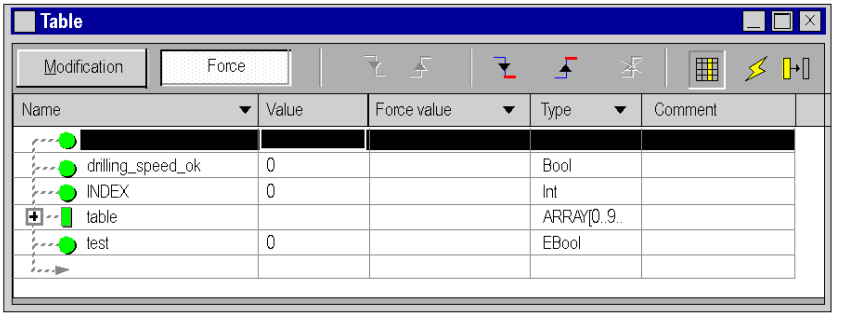
| Step | Action |
|---|---|
| 1 | Click on the  button.<br>The **Set value** column is added to the table:<br>  |
| 2 | In the **Set value** column, modify the value of the Boolean variables you wish.<br>The contents of the corresponding lines is now shown in bold font:<br>  |

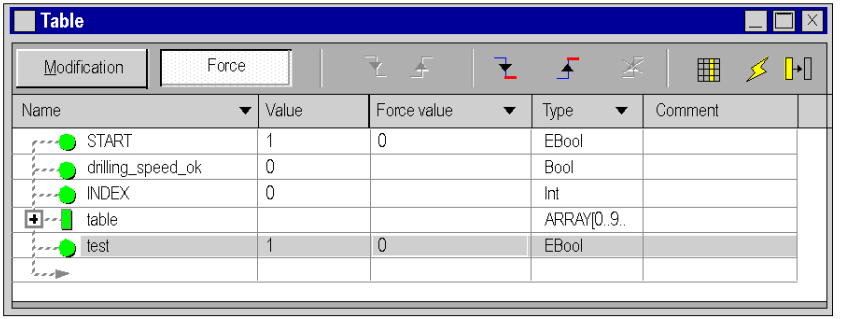
| Step | Action |
|---|---|
| 3 | Press the  button so that the PLC acknowledges the modifications. |

The  button clears the **Set value** column.

## Forcing Several Boolean (EBool) Variables

Carry out the following actions:

| Step | Action |
|---|---|
| 1 | Click on the  button.<br>The **Force value** column is added to the table:<br>  |

| Step | Action |
|---|---|
| 2 | In the **Force value** column, modify the value of the Boolean variables you wish.<br>The contents of the corresponding lines is now shown in bold font:<br>  |
| 3 | Press the  button so that the PLC acknowledges the modifications. |

The  button clears the **Force value** column.

# Animation Table of Forced Bits

## Overview

You can create an animation table that includes all forced bits of a project.

## Table[Forced Bits]

An animation table of forced bits

- can not be modified by you
- is updated automatically if a variable or address becomes forced/unforced or if the value of a forced variable/address changes
- is always created as a temporary animation table and can not be changed into a permanent animation table (see also *Permanent and Temporary Animation Tables, page 1492*)
- has a fixed name (**Table[Forced Bits]**) which can not be modified
- displays the symbolic names of the forced addresses (if mapping is available

## Creating the Animation Table

There are 3 ways to create the animation table of forced bits:

- Via **Tools** → **PLC Screen** open the **PLCScreen** dialog.
  In the **Information** tab under **APPLICATION** → **MISCELLANEOUS** click the **FORCED BITS** line.
- In the **Project Browser** right-click **Animations Tables** and select **Open Forced Bits Table** from the context menu.
- In the status bar click the red **F**.

# Section 39.5
# Instruction Editor (LL984)

## Instruction Editor

### Introduction

For every Unity Pro FFB you can use the Unity Pro animation tables during runtime. (Select a function block and use **Initialize Animation Table** from the context menu. Please refer to *Animation Tables* *(see page 604)*.

But since special FFBs of the Original LL984 Block Library have parameters of array type there will be no detailed information displayed for these parameters.

So for most of the FFBs of the Original LL984 Block Library the LL984 editor provides another means of animation called **Instruction Editor**.

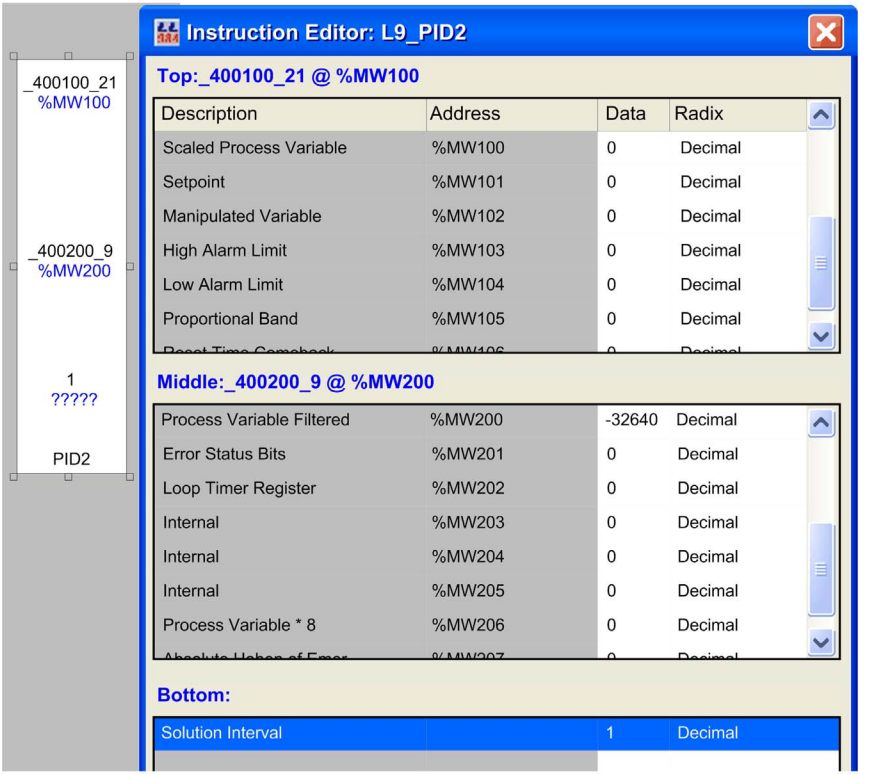
**NOTE:** For the FFBs of the Original LL984 Block Library that are not supported by the **Instruction Editor** see table below.

### Using the Instruction Editor

To open the **Instruction Editor** for an FFB during runtime, select the FFB and use **Show instruction** from the context menu.

## Representation

Representation of the **Instruction Editor**




Items of the **Instruction Editor**

| Item | Description |
|---|---|
| Table Header | displays <Pin name>:<Variable name>@<Address><br>● pin name of the respective function block node<br>● name of the connected variable |
| Description | describes the elements of the array |
| Address | displays the address of the array element in case of located variables<br>In case of unlocated variables this field is empty.<br>In either case the corresponding data column field can be animated/modified. |
| Data | displays the actual value of the array element<br>The value can be edited by clicking this field. |

| Item | Description |
|---|---|
| Radix | allows you to select the displayed data type format<br>Available formats:<br>● Binary<br>● Decimal<br>● Floating Point<br>● HexaDecimal<br>● BCD<br>● ASCII<br>● Signed Integer<br>● 32 Bit Integer |

### Changing the Data Type

If for example an element of the table is defined as of type floating point or 32 bit integer, the next element is shown as disabled.

For some elements you can change the predefined data type.

For example showing a decimal also as binary.

Changing from floating point to decimal will enable the next decimal row.

Changing back to floating point will disable the next decimal row.

### Bit Information

For certain memory words, bit information is also available.

In this case positioning the mouse over the **Data** column of such a memory word will show information for the bits, which may be specially useful when **Binary** is selected as data type format (**Radix**).

### FFBs Not Supported by Instruction Editor

The following FFBs are not supported by the **Instruction Editor**.

| Family | Not Supported FFBs |
|---|---|
| CONTL_COMM | L9_ID<br>L9_IE |
| MISC | L9_LAB<br>L9_RET |

# Section 39.6
# Modifying values:

## Modifying values:

### Introduction

You have the several options for changing values of variables online:

- In the shortcut menu for the variables
  - With the **Change Value**
  - With the command **Set Value**
  - With the command **Force Value**
- In the menu **Services**
  - With the command **Change Value**

This applies to all elementary variables and elementary components of derived variables (DDTs and Arrays) with the following applications:

- In LD (also see the Edge recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )* chapter in the user guide).
  - Contact
  - Coil
  - Operation block
  - Comparison block
- In FBD
  - FFB Pin
- In SFC
  - Transition
- In IL and ST
  - without limitation

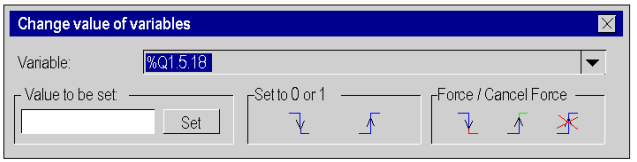
**NOTE:** The above-named commands are only available in the Animation Mode *(see page 1407)*.

## Change value of variables

The **Change value of variables** dialog is modeless. This means that the dialog is not closed after modifying a variable. So you can modify several variables in a section.

One dialog is shown per section. Deactivating a section will hide the opened dialog. Closing a section or stopping the animation of a section will close the dialog.

Appearance:




Depending on the selected variables some functions of the dialog may be grayed out.

Elements:

| Element | Description |
|---|---|
| **Variable** | The selected variable is displayed in this text box.<br>Every variable in a section can be selected either by means of this text box or by clicking on the variable in the section itself. |
| **Value to be set** | Enter the value to be set for the valuable in this text box. When making the entry, ensure the correct syntax of the entered literal, e.g. `30s` for a variable of data type `TIME`.<br>Use **Set** to assign the entered value to the selected variable and close the dialog. |
| **Set to 0 or 1** | This function is only available for Boolean variables.<br>Use  to set the selected variable to 0 and close the dialog.<br>The same function is available with the command **Set Value** → **Set to 0** in the context menu of the variable.<br>Use  to set the selected variable to 1 and close the dialog.<br>The same function is available with the command **Set Value** → **Set to 1** in the context menu of the variable. |

| Element | Description |
|---|---|
| **Force / Cancel Force** | This function is only available for located variables of data type EBOOL.<br>Also see the Edge recognition *(see Unity Pro, Program Languages and Structure, Reference Manual )* chapter in the user guide.<br>Use  to force the selected variable to 0 and close the dialog. (The forced variable will be marked with a frame around the variable name.)<br>The same function is available with the command **Force Value → Set to 0** in the context menu of the variable.<br>Use  to force the selected variable to 1 and close the dialog. (The forced variable will be marked with a frame around the variable name.)<br>The same function is available with the command **Set Value → Force to 1** in the context menu of the variable.<br>Use  to cancel the Force command for the selected variable and close the dialog.<br>The same function is available with the command **Set Value → Cancel Force** in the context menu of the variable. |

# Chapter 40
# Machine-Process Simulator

## Introduction

### Overview

The SIM function allows you to create a virtual machine which simulates the behaviour of all electrical, mechanical, hydraulic components of your process, in the aim to debug the UNITY Pro application before starting up on site.

### Installation

The SIMAC 100 software must be installed separately of Unity Pro software. The software can be downloaded from the PROSYST web site (www.prosyst.fr).

Please, refer to the PROSYST vendor for further installation information.

# Part VII
# Diagnostics

## Subject of this Part

This part describes the various diagnostics tools available in Unity Pro for the programming, maintenance, and use of PLCs.

## What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 41 | Introduction to diagnostics tools | 1533 |
| 42 | Diagnostics screens | 1535 |
| 43 | Diagnostics Viewer | 1541 |
| 44 | System diagnostics | 1561 |
| 45 | Project diagnostics | 1565 |

# Chapter 41
# Introduction to diagnostics tools

## Diagnostics with Unity Pro

### At a Glance

Unity Pro diagnostics consist of a set of tools and functionalities which allow you to intervene at all stages in the development and use of an automation application.

The different resources available are as follows:

- Language Objects (IODDTs *(see Unity Pro, Program Languages and Structure, Reference Manual )*).
- Diagnostics screens.
- The Diagnostics Viewer.
- System diagnostics.
- Project diagnostics (diagnostic EFBs and DFBs).

### Using the resources

The following table shows the resources available.

| Resource | Use |
|---|---|
| Language objects *(see Unity Pro, Program Languages and Structure, Reference Manual )* | Language objects are diagnostics tools which can be used in a project to ascertain the status of a channel or module (for example the error bit for channel 1 of the module located in slot 2 of rack 3: %I3.2.1.ERR). These objects are described at the end of the documentation for each module. In general they are used in the form of IODDT *(see Unity Pro, Program Languages and Structure, Reference Manual )* variables. |
| Diagnostics screens *(see page 1535)* | The diagnostics screens can be accessed from the configuration editor for the processor and the application-specific modules. To access the desired diagnostics screen double-click on the element (processor or module) then select the **Fault** tab. The fault information messages are generally very clear (for example: Range over/undershoot on an analog channel). |
| The Diagnostics Viewer *(see page 1541)* for Unity Pro | The Diagnostics Viewer is a tool which can be used to view the alarms generated by the system diagnostics and the diagnostic DFBs. There is a Diagnostics Viewer built into Unity Pro but there are also Diagnostics Viewers available for Magelis HMI terminals or via Factory Cast applications. |

| Resource | Use |
|---|---|
| System diagnostics *(see page 1561)* | System diagnostics is a service which combines the use of system bits and words with the Diagnostics Viewer. When a project has been **built with the System Diagnostics option** *(see page 519)*, the system bits and words automatically generate diagnostics messages in the Diagnostics Viewers *(see page 1541)*. |
| Project diagnostics *(see page 1565)* (diagnostics EFBs and DFBs and SFC diagnostics) | Project diagnostics consist of specific EFBs and DFBs which are embedded in the code of an automation project to provide the user or maintenance personnel with clear information on the operating conditions of the monitored process. These program elements will trigger alarms which can be viewed in the Diagnostics Viewers. SFC diagnostics go hand in hand with the system diagnostics, which are an intrinsic part of SFC and enable step activity times to be monitored. |

## Advantages

The main advantages of these services are as follows:

- Diagnostics are an integral part of the project. It can therefore be thought out at the development stage in order to be able to best meet the requirements of the user and maintenance personnel.
- Errors are logged and dated at source (in the PLC), meaning that the information provided is an exact representation of the state of the process.
- You can connect several Viewers (Unity Pro, Magelis, Factory Cast) which will describe the exact state of the process to the user. Each Viewer works independently and any action taken with one (an acknowledgment for example) is automatically visible on the others.

# Chapter 42
## Diagnostics screens

# Section 42.1
## Module and processor diagnostics screens

### Subject of this Section

This section introduces the user to the counting module diagnostics screens.

**NOTE: By way of example, only diagnostics for application-specific counting modules are covered here. The diagnostics screens for the other application-specific modules as well as those for processors are similar.**

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Viewing fault diagnostics | 1537 |
| Lists of fault diagnostics | 1539 |

# Viewing fault diagnostics

## At a Glance

The diagnostic screens on the module or channel are only accessible in connected mode. When an **un-masked** fault appears, this is reported:

- in the configuration screen of the rack, with the presence of a red square in the position of the faulty counting module,
- in all screens at module level (**Description** and **Fault** tabs),
  - in the module field with the **I/O** LED.
- in all channel level screens (**Configuration**, **Adjustment**, **Debug** and **Fault** tabs),
  - in the module field with the **I/O** LED.
  - in the channel field with the channel fault LED.
- in the fault screen that is accessed by the **Fault** where the fault diagnostics are described.

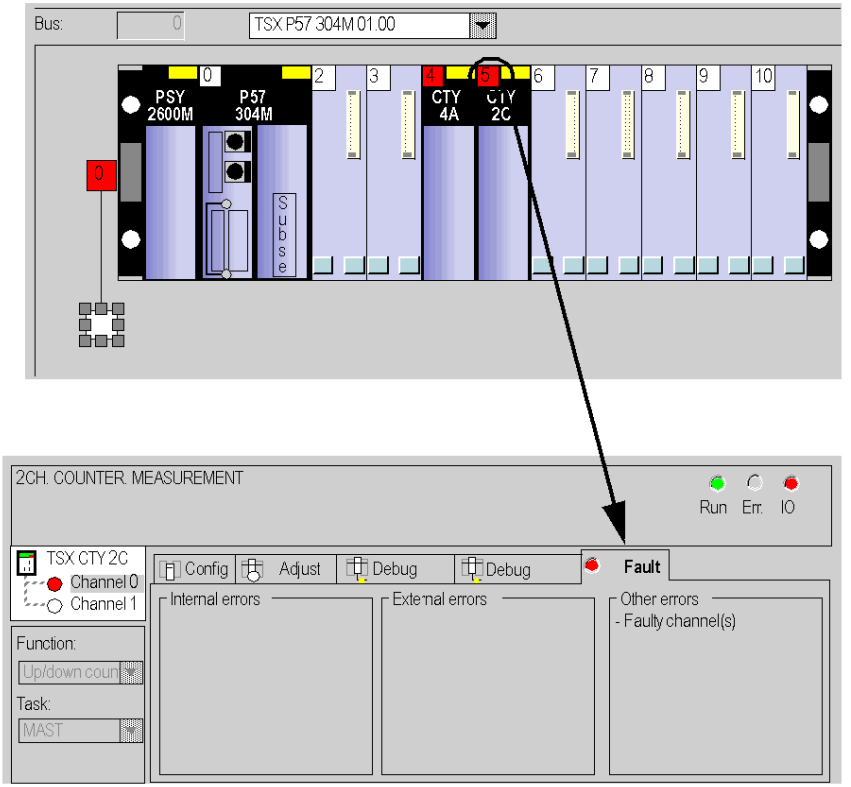
The fault is also reported:

- on the module, through the centralized viewing,
- with the dedicated language objects: **CH_ERROR** (%Ir.m.c.ERR) and module error **MOD_ERROR** (%Ir.m.MOD.ERR), **%MWr.m.MOD.2**, etc., and the status words *(see Premium and Atrium using Unity Pro, Counter modules, User manual)*.

**NOTE:** even if the fault is masked (TSX CTY 2C), it is reported by the flashing of the LED **I/O** and in the fault screen.

## Illustration

The diagram below shows the fault reporting structure.



# Lists of fault diagnostics

## At a Glance

The messages displayed on the diagnostics screens are used to assist in debugging the project. These messages must be concise and are sometimes ambiguous (as different faults may have the same consequences).

These diagnostics are on two levels: module and channel, the latter being the most explicit. The lists below show the message headings with suggestions for finding out what is wrong.

## List of the module fault messages

The table below provides a list of the module fault messages.

| Fault indicated | Possible interpretation and/or action |
|---|---|
| Module failure | The module has a fault.<br>Check the module mounting. Change the module. |
| Faulty channel(s) | One or more channels have a fault.<br>Refer to channel diagnostics. |
| Self-test | The module is running a self-test. Wait until the self-test is complete.<br>**Note: For Modicon M340, if the Xbus line termination are missing, the module will stay in this state.** |
| Different hardware and software configurations | There is a lack of compatibility between the module configured and the module in the rack.<br>Make the configuration and the software configuration compatible. |
| Module is missing or off | Install the module. Fasten the mounting screws. |

## List of channel fault messages

The table below gives the list of fault messages at channel level.

| Fault indicated. Other consequences. | Possible interpretation and/or action. |
|---|---|
| External fault or counting input fault:<br>● encoder or proximity sensor supply fault,<br>● line break or short circuit of at least one encoder differential signal (1A, 1B or 1Z),<br>● SSI series frame fault,<br>● specific fault on absolute encoder.<br>Outputs are set to 0 in automatic mode.<br>**Invalid measurement** message. | Check the sensor connections.<br>Check the sensor power supply.<br>Check the sensor operation.<br>Delete the fault and acknowledge if the fault storing is configured (CTY 2C).<br>Counting pulses or incremental encoder: preset or reset to acknowledge the Invalid measurement message. |
| Counting application fault:<br>● measurement overrun<br>● overspeed<br>Outputs are set to 0 in automatic mode.<br>**Invalid measurement** message. | Diagnose the fault more precisely (external causes).<br>Check the application again, if necessary.<br>Delete the fault and acknowledge if the fault storing is configured (CTY 2C).<br>Counting pulses or incremental encoder: preset or reset to acknowledge the **Invalid measurement** message. |
| Auxiliary input/output fault:<br>● power supply<br>● short circuit of at least one output<br>Outputs are set to 0 in automatic mode. | Check the output connections.<br>Check the input/output power supply (24V).<br>Diagnose the fault more precisely (external causes).<br>Delete the fault and acknowledge if the fault storing is configured (CTY 2C). |
| Internal fault or channel self-test:<br>● faulty module,<br>● module missing or off,<br>● module running self-test. | Module fault has gone down to channel level.<br>Refer to module level diagnostics. |
| Different hardware and software configurations | Module fault has gone down to channel level.<br>Refer to module level diagnostics. |
| Invalid software configuration:<br>● incorrect constant<br>● bit combination not associated with any configuration | Check and modify the configuration constants. |
| Communication fault | Check the connections between the racks. |
| Application fault: refusal to configure or adjust | Diagnose the fault more precisely. |

# Chapter 43
# Diagnostics Viewer

## Subject of this Chapter

The following chapter describes the Diagnostics Viewer, used to display error messages.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Reminder of the Diagnostics Associated with the Viewer | 1542 |
| Operation of the diagnostics in Unity Pro | 1544 |
| Description of the Diagnostics Viewer | 1545 |
| How to find causes | 1548 |
| Configuring the display | 1550 |
| Managing error messages | 1552 |
| Diagnostics Viewer Commands | 1554 |
| DiagViewer Properties | 1556 |
| Archiving error messages | 1558 |

# Reminder of the Diagnostics Associated with the Viewer

## Diagnostics in Unity Pro

In order to check the correct operation of your process, the Unity Pro software offers you 2 types of diagnostics:

- system diagnostics,
- project diagnostics.

The Diagnostics Viewer is used to display error messages.

## System Diagnostics

System diagnostics are performed automatically. When the PLC detects a system error (for example, a watchdog being exceeded, an input/output error, division by zero, etc.), information is sent to the Diagnostics Viewer. The Diagnostics Viewer displays a system error message if in the project settings you have checked the System diagnostics *(see page 519)* box.

## Project Diagnostics

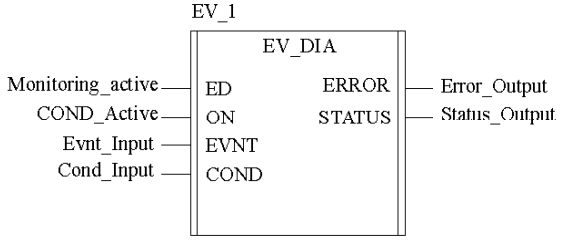
Project diagnostics uses diagnostics EFBs and DFBs. In addition, SFC program diagnostics form an integral part of the system and allow you to test the activation times of the steps.

When the Project diagnostics *(see page 519)* box is checked, information is sent to the viewer when a fault occurs, which displays an error message.

## Diagnostics EFBs and DFBs

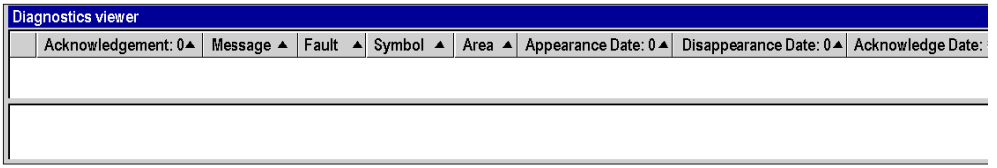
EFBs and DFBs are function blocks that are programmed to carry out diagnostics in your project *(see page 1565)*.

The graphic representation of a DFB is as follows:



## Viewer

The Diagnostics Viewer is a tool that is used to display and acknowledge error messages linked to diagnostics:



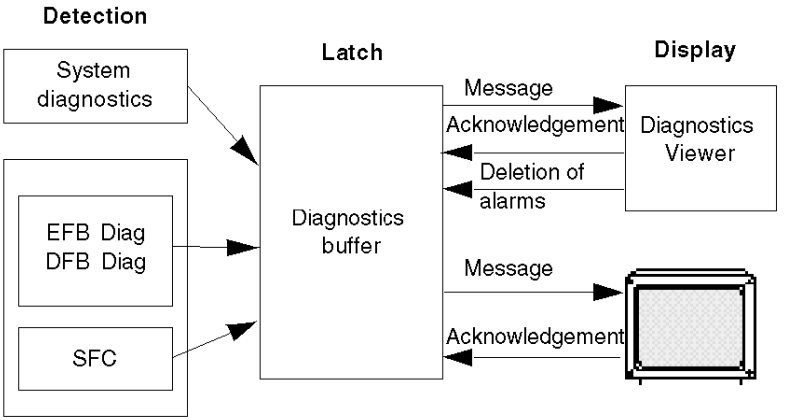
# Operation of the diagnostics in Unity Pro

## General

The Unity Pro diagnostics function monitors the elements of the process to detect any faults there may be. When a fault occurs, it sends an error message to the display systems which display this fault.

## Illustration

The following illustration shows the operation of the Unity Pro diagnostics:




## Operation of the diagnostics

The table below describes the various phases of operation of the Unity Pro diagnostics:

| Phase | Description |
|---|---|
| 1 | The diagnostics function blocks built into the project program and the diagnostics built into the system detect process faults. |
| 2 | The faults are stored in the form of date/time stamped messages in the diagnostics buffer memory. |
| 3 | The error messages are then displayed via the following display systems: Diagnostics Viewer or Human Machine Interface (HMI), used to:<br>● display messages,<br>● acknowledge messages that need an acknowledgement,<br>● initialize animation tables and cross-references,<br>● launch associated program editors,<br>● delete diagnostics buffer alarms. |

# Description of the Diagnostics Viewer

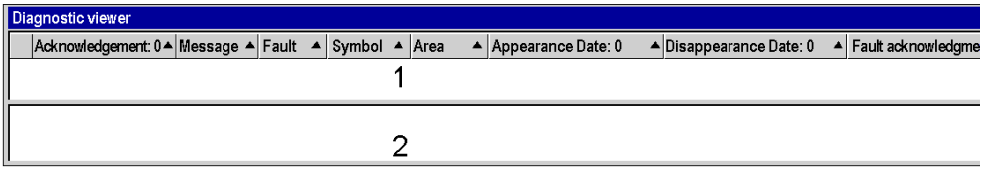
## At a Glance

The Diagnostics Viewer is presented in the form of a display window comprising 2 display fields:

- a field that displays the diagnostics error messages,
- a field that provides additional information on the error message selected from the alarms list.

## Diagnostics Viewer

The following screen presents the Diagnostics Viewer and its 2 display fields:




**1** List of error messages
**2** Additional information on the message selected

## Description of error messages

For each of the alarms, the error messages list displays the following information:

| Field | Description |
|---|---|
| **Acknowledgement** | The icon and the text indicate the message status: not acknowledged, acknowledged, deleted or no acknowledgement. |
| **Message** | Error message. This message contains the 40 first characters of the text entered in the comment of the function block instance that generated the error. |
| **Fault** | Type of diagnostics FB or system bit that detected the error. |
| **Symbol** | Symbol associated with the fault. |
| **Field** | Faulty PLC zone. |
| **Appearance Date** | Date and time of appearance of the fault. |
| **Disappearance Date** | Date and time of disappearance of the fault. |
| **Acknowledge Date** | Date and time of acknowledgement of the fault. |

## Definition of icons

The Diagnostics Viewer uses the following icons to indicate the message status:

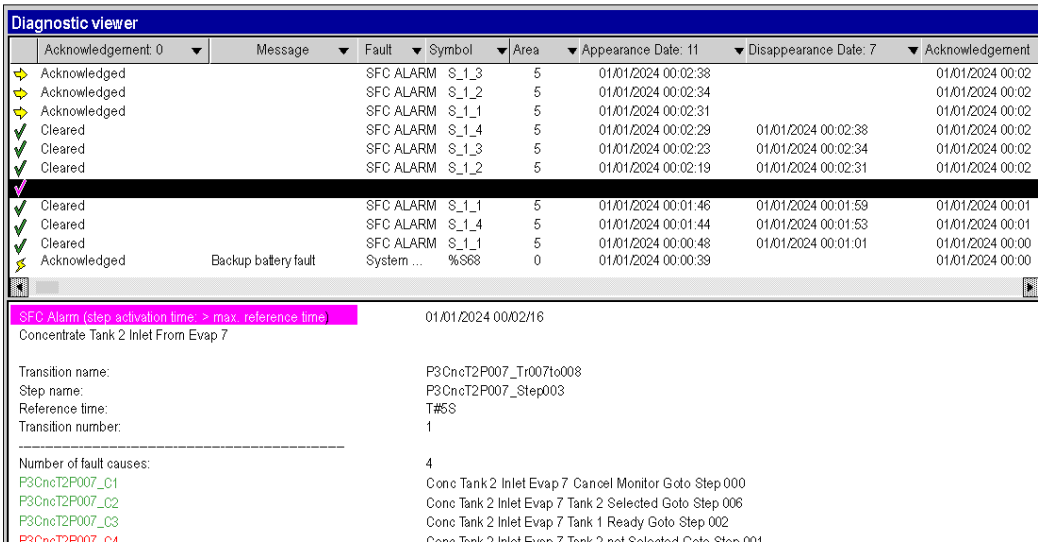
| Icon | Description |
|---|---|
|  | The fault has disappeared and has been acknowledged (if it needed an acknowledgement). |
|  | The fault needs an acknowledgement. |
|  | The fault has not disappeared. |
|  | The system fault has disappeared and has been acknowledged (if it needed an acknowledgement). |
|  | The system fault has not disappeared. |

## Description of the additional information field

The additional information field provides the following information for the message selected in the list:

- the alarm type with
  - a comment (defined by the user for the diagnostics FB)
  - the date of appearance,
- specific variables depending on the type of alarm,
- a list of error variables including any associated comments (if these exist),
- interpretation of the status words as a function of the STO file corresponding to this alarm.

The following screen gives an example of additional information on an error message:




If you want to refresh the values of the status words, reselect the message from the list or press the **F5** function key.

**NOTE:** If you wish, you can find the element that caused the error displayed in the viewer. This is called finding the cause. *(see page 1548)*

# How to find causes

## At a Glance

Cause finding involves searching for an element which has triggered an alarm on a diagnostics EFB or DFB input.

Each input with the **Diag** property can be used to search for causes.

## How to find causes

The following table describes the procedure for finding causes.

| Step | Action |
|---|---|
| 1 | Select the alarm in question in the Viewer.<br>**Result**: The alarm line is shown in reverse video. |
| 2 | Right-click on the input for which you want to search for the cause.<br>**Result**: The contextual menu allows you to initialize an animation table and to search for the DFB input variable so as to ascertain whether it is used in the program. This is called cause finding. |
| 3 | Click on the **Initialize Search** to search for cross-references to the selected input.<br>**Result**: The Cross-References screen is displayed. |

## Lists of diagnostics inputs of diagnostics EFBs or DFBs

The table below specifies the diagnostics inputs (with **Diag** property selected) of diagnostics EFBs or DFBs in the diagnostics library.

| Block name | Block type | Name of diagnostics inputs |
|---|---|---|
| ALRM_DIA | DFB | COND1<br>COND0 |
| D_ACT | EFB | UNLOCK<br>REACT |
| D_DYN | EFB | UNLOCK<br>REACT |
| D_GRP | EFB | IN |
| D_LOCK | EFB | UNLOCK |
| D_PRE | EFB | IN |
| D_REA | EFB | REACT |
| EV_DIA | DFB | EVENT<br>COND |
| MV_DIA | DFB | EVENT<br>COND |

| Block name | Block type | Name of diagnostics inputs |
|---|---|---|
| NEPO_DIA | DFB | REQ_I<br>REQ_O<br>SENSOR_I<br>SENSOR_O<br>SECU_I<br>SECU_O<br>SECUPERM |
| TEPO_DIA | DFB | REQ_I<br>REQ_O<br>SENSOR_I<br>SENSOR_O<br>SECU_I<br>SECU_O<br>SECUPERM |
| SAFETY_MONITOR | DFB | - |
| USER_DIAG_ST_MODEL | DFB | COND |

# Configuring the display

## At a Glance

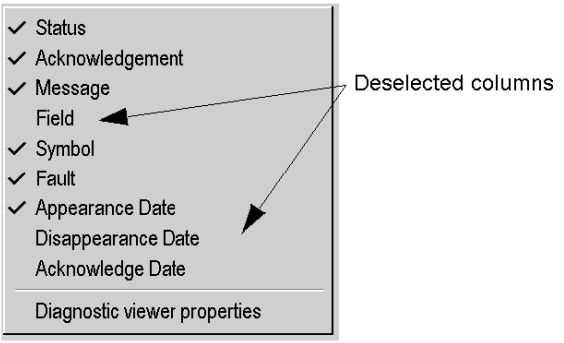
The Diagnostics Viewer allows you to configure the display of the error messages. You can:

- select the columns to be displayed,
- resize the columns,
- choose the color and flashing of the messages.

## Selecting the columns to be displayed

By default the Diagnostics Viewer displays all of its columns, but allows you to customize the display by removing columns that are of no use.

To remove a Diagnostics Viewer column, you must perform the following operations:

| Step | Action |
|---|---|
| 1 | Right-click the title of the column selected.<br>**Result**: A contextual menu containing the list of columns appears on the screen. |
| 2 | Click the title of the column you want to deselect.<br>**Result**: The contextual menu disappears and the column no longer appears in the Diagnostics Viewer.<br>**Note**: To reselect a column you have previously deselected, you must proceed in the same way. In this case, the contextual menu disappears and the column appears in the Diagnostics Viewer. |
| 3 | If you display the contextual menu again (steps 1 and 2), this column will not be preceded by a pointer (deselected column).<br>  |

## Resizing of columns

To modify the size of a column, carry out the following operations:

| Step | Action |
|---|---|
| 1 | Position your cursor in the column titles header, in between the column you want to modify and the next column.<br>**Result**: The column resizing cursor appears on the screen. |
| 2 | When the cursor appears, press and hold the right button of the mouse and move the mouse to change the column size. |
| 3 | Once the correct column size has been reached release the mouse button.<br>**Result**: The column is resized. |

## Choosing the color and flashing of messages

The **Properties** dialog box of the Diagnostics Viewer *(see page 1556)* is used to change the color and flashing of error messages.

# Managing error messages

## At a Glance

The Diagnostics Viewer allows you manage the error messages. You can:

- sort the messages list,
- browse the messages list,
- acknowledge a message in the list,
- delete a message from the list,
- activate another tool,
- delete an alarm from the PLC memory.

## Number of messages

The number of messages displayed in the list is defined solely by the size of the available memory.

If the memory becomes insufficient:

- the DIAG information flashes in the status bar,
- the messages associated with faults that have disappeared and have been acknowledged (if they needed acknowledging) are deleted.

## Sorting messages

You can sort the messages by each column contained in the list except for the one containing the icons.

To sort the messages, simply click the header of the column containing the data you want to sort. If you click the same header a second time it will be sorted in reverse order.

By default the messages are inserted in the list in chronological order of appearance of the faults.

**NOTE:** Whatever the sort criterion for the list, a new message always appears at the **top of the list**.

## Browsing the list

You can browse the list of messages using:

- the **Up**, **Down**, **PageUp**, **PageDown**, **Home** and **End** keys on the keyboard,
- the mouse. Use the scroll bar if the list contains more messages than can be displayed.

### Acknowledging a message

To acknowledge a message that requires acknowledgement, you must select it and activate one of the following commands:

- the corresponding heading in the contextual menu (accessible by right-clicking with the mouse),
- the **F6** function key,
- the corresponding button in the toolbar,
- double-clicking the icon of the alarm selected.

You can acknowledge several messages at once (multiple selection). When you acknowledge a message, an order is sent to the PLC and the associated icon is modified in the list.

A message can be acknowledged from another viewer. In this case, the Unity Pro Diagnostics Viewer is advised and the message is displayed as if acknowledged locally (the associated icon is modified and the message no longer flashes).

### Deleting messages from the list

You cannot delete a message that needs acknowledgement or where the associated fault has not disappeared.

However, you can delete the messages associated with faults that have disappeared and have been acknowledged. To do this activate:

- the corresponding heading in the contextual menu,
- the **Del** key.

### Deleting an alarm from the PLC memory

This function is used to delete from the PLC all the messages selected in the list, even if the fault is still present and has not been acknowledged.

Use of this function is only possible if your user profile *(see page 60)* is different from **ReadOnly** or **Operate**.

### Activating another tool

If at least one message is selected in the list, you can activate the following Unity Pro tools:

- the animation tables for displaying the diagnostics function block data (FB alarms only) and FFB system alarms,
- the cross-references with FB alarms and FFB system alarms only,
- the language editor where the diagnostics function block instance is referenced.

To launch these tools you must activate one of the following commands:

- the corresponding headings in the contextual menu,
- the **Ctrl + T** (animation tables), **Ctrl + U** (cross references) and **Enter** (language editor) function keys,
- the corresponding buttons in the toolbar.

# Diagnostics Viewer Commands

## At a Glance

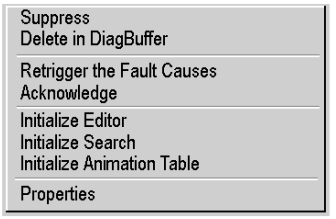
Once you have selected one or more error messages (active errors only), you have a number of options available for accessing the Diagnostics Viewer commands:

- using the contextual menu,
- using the function keys,
- using the toolbar.

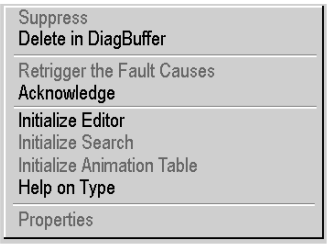
## Contextual Menu

The contextual menu can be accessed by right-clicking a message from the list with the mouse:

None FFB error




FFB error



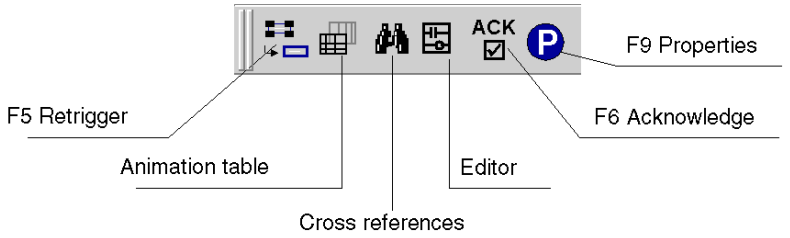

The commands and their function keys from the contextual menu are as follows:

| Command | Function key | Description |
|---|---|---|
| **Suppress** | - | deletion of the message(s) selected |
| **Delete in DiagBuffer** | - | deletion of alarms in the diagnostics buffer |
| **Retrigger the Fault Causes** | F5 | reactivation of the diagnostics function block |
| **Acknowledge** | F6 | acknowledgments of selected message(s) |
| **Initialize Editor** | Enter | activation of the language editor |
| **Initialize Search** | Ctrl + U | activation of the cross references tool (search for active errors) |
| **Initialize Animation Table** | Ctrl + T | activation of the animation table tool (initialize animation table of active errors |
| **Help on Type** | - | opening the online help for the concerned FFB |
| **Properties** | Alt + Enter | displaying the Diagnostics Viewer properties box |

**NOTE:** The **Initialize Search** and **Initialize Animation Table** commands are also accessible through the contextual menu in the additional message information area.

## Toolbar

The icons of the toolbar corresponding to the contextual menu or to the function keys are as follows:



# DiagViewer Properties

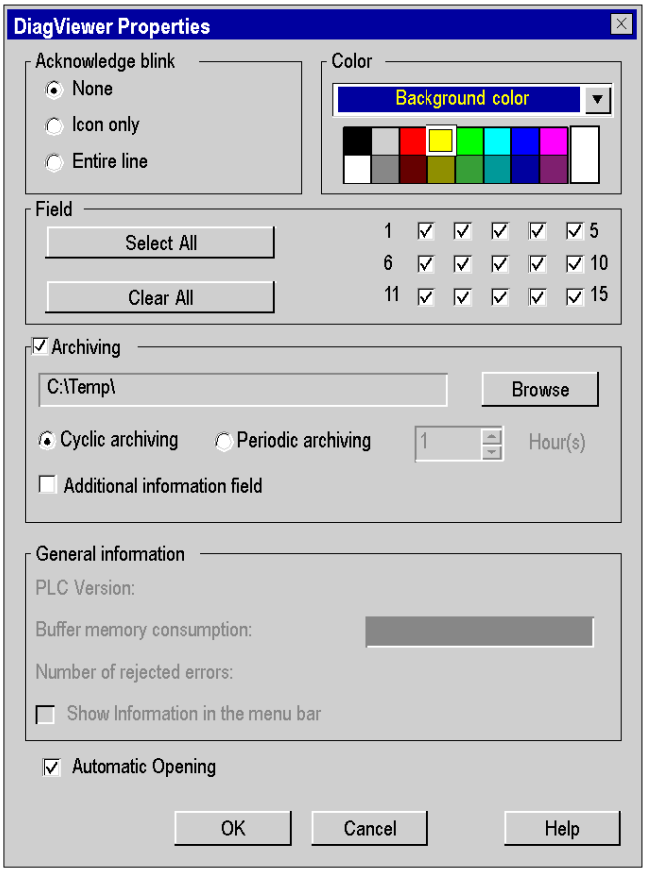
## At a Glance

The **DiagViewer Properties** dialog box, accessible via the **DiagViewer Properties** command in the contextual menu, is used to:

- define the color of the error messages,
- define the flashing mode of the error messages,
- choose the fields to be displayed,
- define the archiving mode,
- define the mode of opening the Diagnostics Viewer.

## Properties dialog box

The screen below presents the **DiagViewer Properties**:



## Description

The following table describes the various fields of the **Properties** dialog box:

| Field | Description |
|---|---|
| **Acknowledge blink** | Allows you to define the type of flashing when an error message to be acknowledged appears. Once acknowledged messages no longer flash.<br>● **None**: no flashing,<br>● **Icon only**: only the icon flashes,<br>● **Entire line**: the entire line flashes. |
| **Color** | Allows you to define the color of the error messages. |
| **Field** | Allows you to choose the fields to be displayed (fields 0 to 15). By default all the fields are displayed.<br>● **Select all**: selects all fields,<br>● **Clear All**: deselects all fields. To select a field check the corresponding box by clicking the mouse. |
| **Archiving** | Click this box to generate an archive *(see page 1558)*. The entry field and the **Browse** button allow you to define the path of the archive file.<br>● **Cyclic archiving**: a new archive file is generated every 1000 saves.<br>● **Periodic archiving**: a new archive file is generated every x hours.<br>● **Hour** allows you to define the number x (between 1 and 24).<br>● **Additional information field**: allows you to store additional data in the file. |
| **general information** | Displaying various information:<br>● PLC version.<br>● diagnostics buffer memory usage.<br>● number of alarms rejected if the diagnostics buffer is full.<br>● **Show Information in the menu bar** is used to display this information in the title bar of the diagviewer window. |
| **Automatic Opening** | When this button is checked, the Diagnostics Viewer is automatically displayed when a fault appears. |

# Archiving error messages

## At a Glance

Archiving is used to store in a file all error messages as soon as they appear, disappear or are acknowledged. As soon as a message is read in the diagnostics buffer it is saved in the archive file.

If a fault appears and then disappears, it is represented by a single message in the Diagnostics Viewer (a single line), but by 2 lines in the archive file.

The **Properties** dialog box allows you to activate archiving of error messages. It also allows you to define the type of archiving and the location of the archive file.

## Type of archiving

You can choose between 2 types of archiving:

- **cyclic archiving**: The history file created is named *ProjectName.his*, were *ProjectName* is the name you entered in the **Project Properties** dialog box is.
  in order to avoid the size of the archive file becoming too large, a new *ProjectName.his* file is created every 1000 saves. The old file then changes name and becomes *ProjectName.bak*.
  **Note:** If a *ProjectName.bak* file already exists, it will be overwritten by the new file of the same name. To avoid a loss of data, you must therefore manage your archive files.
- **periodic archiving**: the user defines the frequency with which a new archive file is created (between 1 and 24 hours). The file is created under the name *ProjectDraft_date_time* (for example, for a file created on 31 July 2001 at 12 h 26 min 55 s: *ProjectDraft_073101_122655*. If the chosen frequency is 1 h, the next file will be created under the name *ProjectDraft_073101_132655*, etc.). Creating a new file does not overwrite the previous file.

## Archive file format

The archive file is a delimited ASCII file (each item of information is separated by the **;** character). You can therefore easily import this file into a text editor or a spreadsheet.

Each line of the file takes the following form:

```
SystemAlarm ; 256 ; 1 ; 51 ; 0 ; 28/04/03  23:50:13 ; 28/04/03 23:51:15;
28/04/03 23:57:16 ; 0 ; 2; 10003 ;148 ; Arithmetic error ; %S18 (MAST);
```

where:

- **SystemAlarm**: identifies the fault.
  The types of fault can be:
  - FBAlarm: function block fault
  - SFCAlarm: sequential function chart fault
  - SystemAlarm: system fault
  - UndefAlarm: undefined fault
  - AlarmERROR: errors
- **256**: fault identifier,
- **1**: name of this fault,
- **51**: state of the diagnostics bits,

- **0**: number of the declared area,
- **28/04/03 23:50:13**: date and time of appearance of the fault,
- **28/04/03 23:51:15**: date and time of acknowledgement of the fault,
- **28/04/03 23:57:16**: date and time of disappearance of the fault,
- **0**: available number of reasons for the fault,
- **2**: status length of the fault,
- **10003**: fault type,
- **148**: fault classification,
- Arithmetic error; %S18 (MAST): additional data if you have checked the **Additional information field** box.

The following table describes the additional information to a message according to fault type.

| Error type | Information |
|---|---|
| FBAlarm | Comment<br>Name of the function block instance<br>Name of the function block type<br>Name of the input or output of the faulty block |
| SFCAlarm | Comment<br>Name of the faulty step<br>Name of the faulty transition<br>Format of the reference time<br>Additional information |
| SystemAlarm | Fault name<br>Additional information according to the fault:<br>● watchdog,<br>● arithmetic,<br>● task,<br>● SFC,<br>● ... |

# Chapter 44
# System diagnostics

## System diagnostic

### At a Glance

The system diagnostic is performed automatically. When the PLC detects a system error (for example, a watchdog being exceeded, an input/output error, division by zero, etc.), information is sent to the diagnostic viewer. The diagnostic viewer will display a system error message if you have checked the System diagnostics *(see page 519)* checkbox.

**NOTE:** As with the project diagnostic, the information viewed on the Viewer comes from the PLC's diag buffer *(see page 1544)*, consequently the events are dated at the source and give the precise status of the process monitored.

### Implementation

The following table describes the procedure to follow to use the system diagnostic on the Premium, Atrium and Quantum PLCs in Unity Pro.

| Step | Action |
|---|---|
| 1 | Select **Tools** → **Project Settings**<br>Result: the project settings configuration window is displayed. |
| 2 | In the **Diagnostics** area on the **Build** tab, check the **System diagnostics** *(see page 519)* checkbox. |
| 3 | Confirm with **OK**. |
| 4 | Build your project. |
| 5 | Transfer your project to the PLC. |
| 6 | Open the Diagnostic Viewer by selecting: **Tools** → **Diagnostic Viewer**.<br>**Result**: the system diagnostic is operational and each system alarm now appears in the Viewer. |

## List of system alarms monitored

The following table summarizes the system information monitored automatically by the system diagnostic service.

| System object | Succinct description of the alarm |
|---|---|
| %S10 | Input/output error |
| %S11 | Watchdog overflow! |
| %S15 | Character string fault |
| %S18 | Overflow or arithmetic error |
| %S19 | Task period overrun |
| %S20 | Index overflow |
| %S39 | Saturation in event processing |
| %S51 | Time loss in real time clock |
| %S65 | Extract card command |
| %S66 | Backup application to the memory card |
| %S67 | State of the PCMCIA application memory card battery |
| %S68 | State of processor battery |
| %S76 | Diagnostic buffer configured |
| %S77 | Diagnostic buffer full |
| %S96 | Previously backup program |
| %S118 | General Fipio I/O fault |
| %S119 | General in-rack I/O fault |
| %SW0 | Master task scanning period |
| %SW1 | Fast task scanning period |
| %SW2 | Period of auxiliary task scanning 0. |
| %SW3 | Period of auxiliary task scanning 1. |
| %SW4 | Period of auxiliary task scanning 2. |
| %SW4 | Period of auxiliary task scanning 3. |
| %SW11 | Watchdog duration |
| %SW17 | Error status for floating operation |
| %SW76 | Diagnostic function: save |
| %SW77 | Diagnostic function: de-registration |
| %SW78 | Diagnostic function: number of errors |
| %SW96 | Save/restore %MW in flash memory. |

| System object | Succinct description of the alarm |
|---|---|
| %SW97 | Storage card error code |
| %SW125 | Type of blocking error |
| %SW146 | Fipio bus arbiter function display |
| %SW153 | List of Fipio channel manager faults |
| %SW154 | List of Fipio channel manager faults |

**NOTE:** The Fipio diagnostic is integrated in Unity Pro versions higher than 1.0.

# Chapter 45
# Project diagnostics

## Subject of this Chapter

This chapter describes how to implement project diagnostics and its functionalities.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| How to Create a User Diagnostic DFB | 1566 |
| Project diagnostic | 1568 |

# How to Create a User Diagnostic DFB

## At a Glance

You can create a diagnostic DFB in one of two ways.

- You can create a standard DFB and then define its values and program it so that it is a diagnostic DFB.
- You can use the USER_DIAG_ST_MODEL *(see Unity Pro, Diagnostics, Block Library)* model that we will provide you with, or use a model you already have.

These two procedures are described below.

**NOTE:** to understand how user diagnostic DFBs work, refer to the USER_DIAG_ST_MODEL *(see Unity Pro, Diagnostics, Block Library)* model described in the diagnostic library.

## Creating a Diagnostic DFB without a Model

The following table describes the procedure to follow to create a diagnostic DFB without a predefined model.

| Step | Action |
|---|---|
| 1 | In the structure view of the Project Browser, right-click on the directory **Derived FB Types**.<br>**Result**: A shortcut menu is opened. |
| 2 | Activate the **Open** command.<br>**Result**: The Data Editor window is opened. It is positioned directly under the **DFB Types** tab and shows the list of the project's DFBs. |
| 3 | Select the first empty cell in the **Name** column (indicated by an arrow) and enter the name of the new DFB type, then confirm using **Enter**.<br>**Result**: The empty structure of the DFB type is now created. This new DFB is added to the list of existing DFBs. It also appears in the **Derived FB Types** directory tree. |
| 4 | In the Data Editor, right-click on the type of DFB you have just created.<br>**Result**: the contextual menu appears. |
| 5 | Choose the **Data Properties** command.<br>**Result**: the Data Properties window appears. |
| 6 | Click on the **Diag** button.<br>**Result**: the checkbox is checked in red. This type of DFB is now a diagnostic DFB. If, in the Project Settings *(see page 519)*, the application diagnostic has been requested, the project's generation will incorporate the project diagnostic service in the PLC. |
| 7 | Create the inputs, outputs, inputs/outputs and the internal variables of the DFB.<br>**Note**: in the input's properties, remember to check the **Diag** checkbox for inputs that will be monitored by the DFB. |

| Step | Action |
|---|---|
| 8 | Write your DFB's code using the REGDFB *(see Unity Pro, Diagnostics, Block Library)* function to save and date the alarm, and the DEREG *(see Unity Pro, Diagnostics, Block Library)* function to de-register the alarm.<br>**Note**: refer to the description of the code of the *(see Unity Pro, Diagnostics, Block Library)*USER_DIAG_ST_MODEL user diagnostic DFB model if you are not familiar with this type of DFB. |
| 9 | Run a project analysis to validate your new type of DFB.<br>**Result**: your new type of DFB is created. You can now create and insert instances of this type of DFB in your program. When your PLC is in RUN mode, the Viewer *(see page 1541)* will display the alarms according to the operation you have programmed. |

## Creating a Diagnostic DFB with the USER_DIAG_ST_MODEL Model

The following table describes the procedure to follow to create a diagnostic DFB using the USER_DIAG_ST_MODEL predefined model.

| Step | Action |
|---|---|
| 1 | In the structure view of the Project Browser, right-click on the directory **Derived FB Types**.<br>**Result**: A shortcut menu is opened. |
| 2 | Choose the **Get from Library** command.<br>**Result**: the **Library Browser** window appears. |
| 3 | Select the USER_DIAG_ST_MODEL diagnostic DFB model in the diagnostic library, then click on the **>>** button to transfer it to your project. |
| 4 | Confirm the transfer with **OK**.<br>**Result**: the model appears in the Project Browser. |
| 5 | Replace the USER_DIAG_ST_MODEL name with the name of the type of user DFB you want to create. |
| 6 | Modify and create the inputs, outputs, inputs/outputs and the internal variables of the DFB.<br>**Note**: in the input's properties, remember to check the **Diag** checkbox for inputs that will be monitored by the DFB. |
| 7 | Write your DFB's code using the model.<br>**Note**: refer to the description of the code of the user diagnostic DFB model *(see Unity Pro, Diagnostics, Block Library)* if you are not familiar with this type of DFB. |
| 8 | Run a project analysis to validate your new type of DFB.<br>**Result**: your new type of DFB is created. You can now create and insert instances of this type of DFB in your program. When your PLC is in RUN mode, the Viewer *(see page 1541)* will display the alarms according to the operation you have programmed.<br>**Note**: the message displayed by the Viewer is the comment associated with the instance of the DFB. You must therefore remember to fill in this comment for each instance created. |

# Project diagnostic

## At a Glance

The project diagnostic uses the diagnostic EFBs and DFBs and the diagnostic integrated in the SFC to generate alarms on the Diagnostic Viewer.

Each diagnostic EFB and each diagnostic DFB has its own specific operation described in the diagnostic library *(see Unity Pro, Diagnostics, Block Library)*. If you cannot find the appropriate EFB or DFB among these elements, you can create your own diagnostic DFB *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

**NOTE:** It is strongly recommended to only program a diagnostic DFB instance once within the application.

**NOTE:** As with the system diagnostic, the information viewed on the Viewer comes from the PLC's diag buffer *(see page 1544)*, consequently the events are dated at the source and give the precise status of the process monitored.

## Implementation of diagnostic EFBs or DFBs

The following table describes the procedure to follow to use the project with a diagnostic EFB or DFB on the Premium, Atrium and Quantum PLCs in Unity Pro.

| Step | Action |
|---|---|
| 1 | Select **Tools** → **Project Settings**<br>Result: the project settings configuration window is displayed. |
| 2 | In the **Diagnostics** area on the **Build** tab, check the **Application diagnostics** *(see page 519)* checkbox. |
| 3 | Choose the language of the messages in the Viewer. |
| 4 | Choose the **Application level** *(see page 519)* (alarm cause search level). |
| 5 | Confirm with **OK**. |
| 6 | Integrate the diagnostic EFBs or DFBs *(see Unity Pro, Diagnostics, Block Library)* in your application.<br>**Note**: The messages displayed in the Viewer will be the comments you will have associated with the instances of your diagnostic EFBs or DFBs. |
| 7 | Build your project. |
| 8 | Transfer your project to the PLC. |
| 9 | Open the Diagnostic Viewer by selecting: **Tools** → **Diagnostic Viewer**.<br>**Result**: the system diagnostic is operational and each alarm generated by your EFBs or DFBs now appears in the Viewer. |

## Implementation of the SFC diagnostic

The following table describes the procedure to follow to use the SFC diagnostic on the Premium, Atrium and Quantum PLCs in Unity Pro.

| Step | Action |
|---|---|
| 1 | Select **Tools** → **Project Settings**<br>Result: the project settings configuration window is displayed. |
| 2 | In the **Diagnostics** area on the **Build** tab, check the **Application diagnostics** *(see page 519)* checkbox. |
| 3 | Confirm with **OK**. |
| 4 | Build your project. |
| 5 | Transfer your project to the PLC. |
| 6 | Open the Diagnostic Viewer by selecting: **Tools** → **Diagnostic Viewer**.<br>**Result**: the SFC diagnostic is operational and each alarm linked to the SFC now appears in the Viewer. |

## Error message display

The number of messages it is possible to display is limited only by the size of the memory buffer.When there is not enough memory, a message warns the user and any messages of errors that have disappeared or have been acknowledged (if necessary) are then deleted.

It is possible to modify the color of the messages and the blinking associated with an acknowledged message.

In the viewer, it is possible to show only those messages which come from one or more specific zones.

The list of messages can be sorted according to each field. To do this, simply click on the column header containing the data on the basis of which the sort is to be carried out.

A second click carries out the sort in opposite order.

By default, the error messages are inserted into the list in the chronological order in which they appear.

# Part VIII
## Operator screen

### Subject of this Part

This part describes the operator screens built into Unity Pro:

### What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 46 | General information | 1573 |
| 47 | First steps | 1577 |
| 48 | Editing runtime screens | 1589 |
| 49 | Runtime screens in online mode | 1667 |
| 50 | Runtime screens objects library | 1679 |

# Chapter 46
# General information

## Object of this section

This chapter introduces Unity Pro ’s built-in operator screens: description, creation and operator screens

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Introduction | 1574 |
| Creation of Runtime Screens | 1575 |

# Introduction

## Introduction

### Introduction

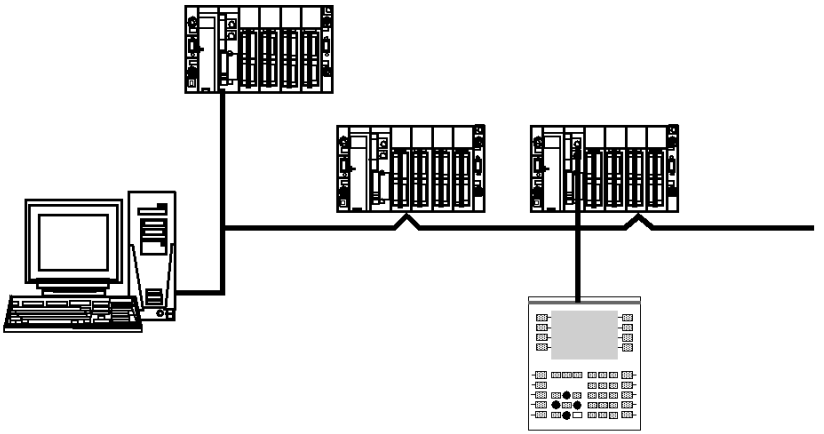
The operator screens aid the operating of an automated process.

They use in the Unity Pro software:

- The **project browser** for browsing through the screens and launching different tools (the graphics editor, variables editor, messages editor, ...)
- The **graphics editor** for creating or changing screens. In connected mode, it also allows the viewing of animated screens and process driving
- The **library of objects** which presents design objects and enables their insertion in the screens. It also allows users to create their own objects and insert them in a library family.

### Example of use

Example of automation structure that uses operator screens:



In this structure, we find:

- The **PLC** which contains the process management automation project.
- The **terminal** which contains the automation project with the operator screens. It is connected to the PLCs by the console line or by a network.
- The operator screens view the process and can be ordered by the keyboard of the **PC**, the mouse or even by a command **terminal** connected to the PLCs.

# Creation of Runtime Screens

## Introduction

For a given project, you can use the graphic editor to create runtime screens.

These screens are produced with texts and graphic objects that you can draw yourself (lines, rectangles, curves, etc.) or retrieve from the graphic objects library. They comprise static (background, title, etc.) and dynamic (animated) parts which reflect the status of the process.

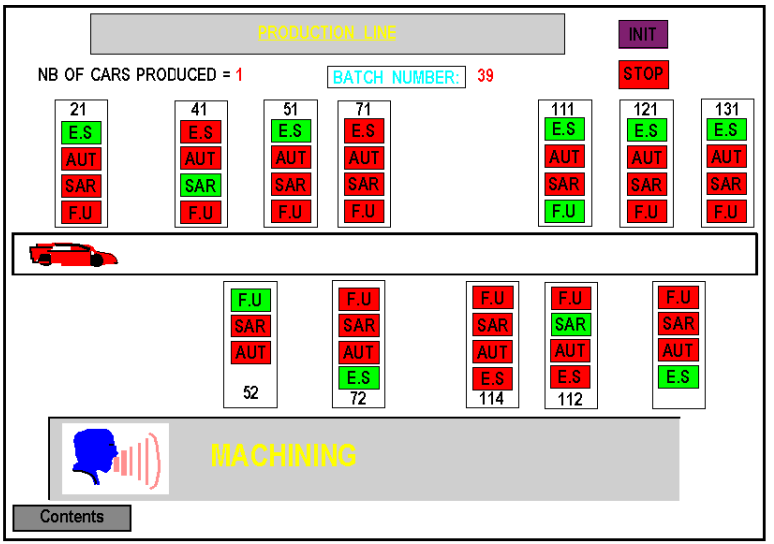
You must assign a variable to them in order to animate the dynamic objects, the variables' value will determine display.

You can also insert driving objects (buttons, input zones, etc.) in your screens to drive the process.

The screens can then be linked together in order to respond to the specific automation requirements.

## Example of screen

Example of a screen that shows the state of a process, the quantities produced and allows the safety management of a **production line**.This screen comprises dynamic displays which change to match the process and static displays that remain unchanged.



# Chapter 47
# First steps

## Object of this chapter

This chapter gives the elements required for understanding the built-in runtimescreens: notion of runtime screen editor, main menus, etc.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 47.1 | Working principles | 1578 |
| 47.2 | Menus and tools in the runtime screen editor | 1580 |
| 47.3 | Configuring options from the runtime screen editor | 1587 |

# Section 47.1
# Working principles

## How to access the operator screens

### At a Glance

An operator screen is a window just as any other editor (configuration, language, data editor). This window can be opened, moved, and resized.

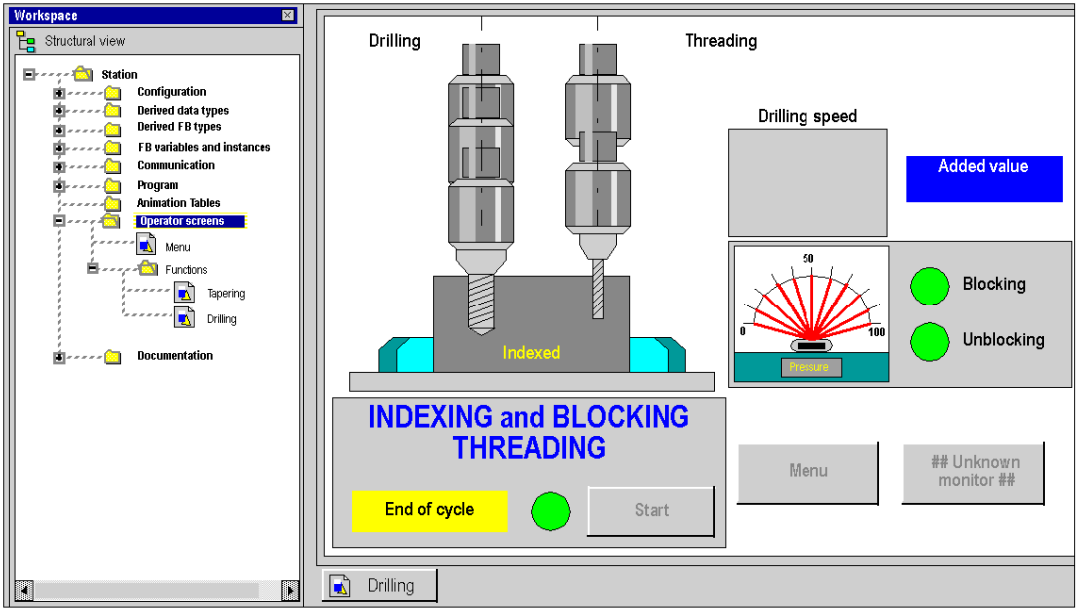
### Methodology

The following table lists the operations to perform in order to access operator screens.

| Step | Action |
|---|---|
| 1 | View the project in structural view (**View** → **Structural View**). |
| 2 | In the project browser, expand the folder **Operator Screens** or the folder of the family for which you want to open a screen. |
| 3 | Open the screen or screens to be viewed. Select a screen to open it and activate the command **Open** in the contextual menu. |

## Example of screen

The following screen shows an example of operator screens.



# Section 47.2
## Menus and tools in the runtime screen editor

### Subject of this section

This section introduces the different menus as well as the toolbars offered by the runtime screen editor.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Menus in the operator screen editor | 1581 |
| Edit toolbar | 1584 |

# Menus in the operator screen editor

## Main menus in the operator screens

The Unity Pro menu bar is identical for all tools accessible from the project browser. This is a contextual bar; in other words the commands accessible via the menus depend on the element (window, graphic object, etc.) that is activated when the menu is scrolled down.

The commands of the operator screen editor can be accessed via the menus:

- **Edit** which enable the creation or modification of the operator screens,
- **View** is used to define screen properties (background color, size, etc.),
- **Services** for choosing the editing tool, selecting an animated object, etc.

## Edit menu

Commands that can be accessed in the **Edit** menu are.

| Field | Description |
|---|---|
| **Undo all** | Undoes all changes made in the screen since the last validation. |
| **Enable** | Validates all changes made in the screen since the last validation. |
| **Cancel** | Cancels the last completed change. |
| **Redo** | Re-performs the changes previously cancelled by a command **Cancel**. |
| **Cut** | Deletes from the screen the selected item(s) and places it (them) on the clipboard. |
| **Copy** | Copies the selected item(s) and places it (them) on the clipboard. |
| **Paste** | Pastes the content of the clipboard on the screen. |
| **Delete** | Deletes from the screen the selected item(s), without copying them in the clipboard. |
| **Select all** | Selects all the items contained in the screen |
| **New** | Sub-menu from which you may select:<br>**Line**: Create a line<br>**Rectangle**: Create a rectangle<br>**Ellipse**: Create an ellipse<br>**Curve**: Create a curve<br>**Polygon**: Create a closed polygon<br>**Text**: Create texts<br>**Image**: incorporation of an image<br>**Screen browsing**: Create an inter-screen browser button<br>**Controls**: Creating a driving object (command) of type:<br>● button<br>● checkbox<br>● entry field<br>● counter<br>● cursor<br>● explicit exchanges |

| Field | Description |
|---|---|
| **Select mode** | Select one or several objects. |
| **Align** | Sub-menu from which you may select:<br>**Top**: Upper alignment<br>**Bottom**: Lower alignment<br>**Left**: Left alignment<br>**Right**: Right alignment |
| **Flip** | Sub-menu from which you may select:<br>**Flip Vertical**<br>**Flip Horizontal** |
| **Group** | Grouping of objects. |
| **Ungroup** | Ungrouping of objects. |
| **Order** | Sub-menu from which you may select:<br>**Foreground**: Placing in the foreground<br>**Background**: Placing in the background |
| **Properties** | Is used to define properties of the selected item: color, frame, piloted variable, related text, etc. |
| **Information** | Displays information specific to the selected object: type of item, position, size, related variable. |

### Display menu

Commands accessible from the **View** menu are as follows.

| Field | Description |
|---|---|
| **Full screen** | Full screen display of the graphics editor window. |
| **Open model used** | Enables user to open the model if it is used in the current screen. |
| **Screen properties** | Is used to define the properties of the active screen: screen name, background color, family, etc. |

## Services Menu

The commands accessible from the **Services Menu** areas follows.

| Field | Description |
|---|---|
| **Animation** | Starts or stops the animation of the operator screen |
| **Enable variable modification** | Authorizes the user to modify the variables in the PLC from the command items. |
| **Driving via the PLC** | Used to drive the screens with the PLC. The PLCs requirements are taken into account by means of a variable defined in the advanced parameters *(see page 550)*.<br>Since the screen is called up by the PLC, editing and modification of the configuration parameters are disabled. |
| **Driving via the operator** | Used to drive the screens via the Operator. The requirements of the PLC are disabled, it is the operator who selects the screen that he wants to view. |
| **Previous animated object** | Selects the previous animated object. |
| **Next animated object** | Selects the next animated object. |
| **Initialize search** | Used to create a list of cross references for variables associated with objects selected. |
| **Initialize animation table** | Used to create an animation table made up of the variables associated with selected items. |

## Important

Right clicking can be used to gain direct access to the functions available for the current context. This is the contextual menu.

In online mode, all commands are available if they have not been disabled *(see page 563)*.

## Multi-instantiation

The operator screen editor is used to open several windows (instances) to open simultaneously. Each window contains a different screen. The keyboard command **Ctrl + Tab** is used to switch between windows. Even if several windows are open, only a single tool bar is displayed and acts on the active window alone.

The full screen display is only used to view one screen.

# Edit toolbar

## At a Glance

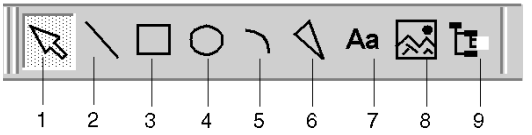
Clicking **Tools** → **Customize** displays the Edit toolbar described above. Each icon (tool) corresponds to an action concerning one or several objects in the graphical editor.

The figure below presents the tools available via the editor toolbar.



## Drawing tools for standard objects

The available tools are as follows:

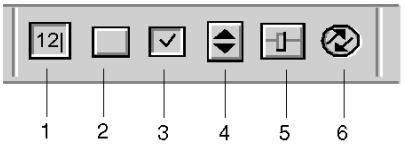



They are used to complete the operations below:

| Number | Description |
|---|---|
| 1 | Select one or more objects |
| 2 | Create a line |
| 3 | Create a rectangle |
| 4 | Create an ellipse |
| 5 | Create a curve |
| 6 | Create a closed polygon |
| 7 | Create texts |
| 8 | Incorporation of an image |
| 9 | Create an inter-screen browser button |

## Design tools for command objects

The available tools are as follows:

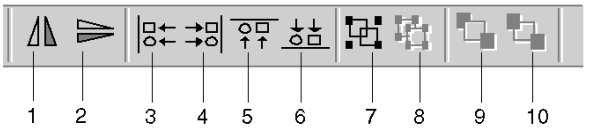


They are used to complete the operations below:

| Number | Description |
|---|---|
| 1 | Create an input zone |
| 2 | Create a button |
| 3 | Create a check box |
| 4 | Create a counter |
| 5 | Create a cursor |
| 6 | Create an explicit exchange button |

## Positioning tools for a group of objects

The available tools are as follows:




They are used to complete the operations below:

| Number | Description |
|---|---|
| 1 | Flip Vertical |
| 2 | Flip Horizontal |
| 3 | Left alignment |
| 4 | Right alignment |
| 5 | Upper alignment |
| 6 | Lower alignment |
| 7 | Grouping of objects |
| 8 | Ungrouping of objects |
| 9 | Placing in the foreground |
| 10 | Placing in the background |

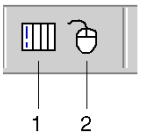
## Object properties

The proposed tool is the following. It is used to access the properties of an object.



## Screen driving tools

The available tools are as follows:

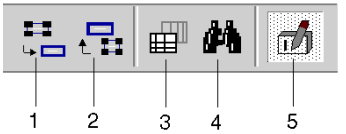



They are used to complete the operations below:

| Number | Description |
|---|---|
| 1 | Driving via the PLC. This service may also be accessed via a keyboard shortcut **F2**. |
| 2 | Driving via the operator. This service may also be accessed via a keyboard shortcut **F3**. |

## Access tools to the animated objects

The available tools are as follows:




They are used to complete the operations below:

| Number | Description |
|---|---|
| 1 | Selects the next animated object. This service may also be accessed via a keyboard shortcut **F4**. |
| 2 | Selects the previous animated object. This service may also be accessed via a keyboard shortcut **F5**. |
| 3 | Initialization of an animation table. This service may also be accessed via a keyboard shortcut **CTRL+T**. |
| 4 | Creates a list of cross-references for variables associated with objects selected. This service may also be accessed via a keyboard shortcut **CTRL+U**. |
| 5 | Authorizes the user to modify the variables in the PLC from the command items. This service may also be accessed via a keyboard shortcut **F7**. |

# Section 47.3
# Configuring options from the runtime screen editor

## General configuration of operator screens

### At a Glance

General configuration options and settings for all the operator screens can be used to customize their functions.

The options *(see page 563)* are accessible by clicking **Tools → Options → Operator Screens tab**.

The settings *(see page 550)* are accessible by clicking **Tools → Project Settings → Operator Screens tab**.

These modifications are immediate and are automatically saved.

# Chapter 48
# Editing runtime screens

## Object of this section

This chapter describes the different functions offered to the user, for editing the runtime screens: create, modify, delete a screen or a family of screens, manage built-in runtime screen messages etc.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 48.1 | Editing screens from the browser | 1590 |
| 48.2 | Creating an object | 1602 |
| 48.3 | Default Properties screen | 1614 |
| 48.4 | Modification of an object's attributes | 1616 |
| 48.5 | Manipulation of objects in runtime screens | 1645 |
| 48.6 | Managing the runtime screens messages editor | 1655 |

# Section 48.1
## Editing screens from the browser

### Subject of this section

This section describes the different editing actions that users can perform on the screens or family of screens: create, modify, delete, etc.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Editing operator screens | 1591 |
| Operator screen editor functions | 1593 |
| Screen Family Editor Functions | 1596 |
| Operator screen properties | 1598 |
| Properties of Operator Screen Families | 1601 |

# Editing operator screens

## Introduction

You can access operator screens through the project browser in the directory **Operator Screens**. It displays the information linked to each project in a tree directory.

The screens may be directly associated with the directory or grouped in **families** of screens.

The tree directory of operator screens may be expanded and contracted in order to obtain the required level of display.

## Screens

The operator screens accept all editing functions:

- Create
- Modify
- Delete
- Move
- Attach/Detach
- Export

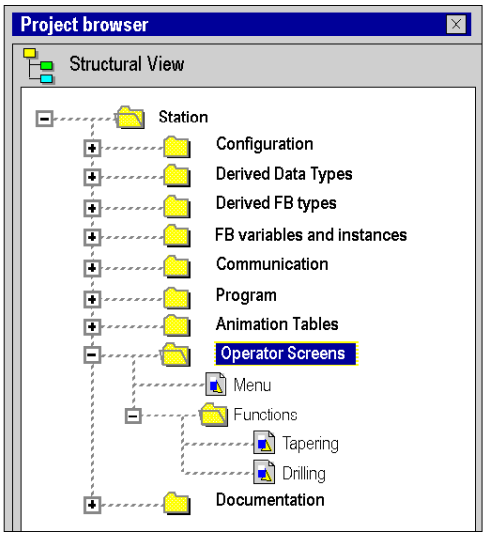
## Screen families

The screens are regrouped in families which accept the same editing functions as the screens:

- Create
- Modify
- Delete
- Move
- Attach/Detach
- Export

## Introduction to the project browser window

The screen below shows the project browser window with in this example a single screen (Menu) and a screen family (Functions).



# Operator screen editor functions

## At a Glance

You can access all functions via the **Edit** menu or via the context menu by right clicking in the project browser window.

## How to Create a Screen

The Table Below Shows the Procedure for Creating a Screen.

| Step | Action |
|---|---|
| 1 | Right click the **Operator Screens** directory contained in the project browser tree directory.<br>**Result:** The relevant contextual menu appears |
| 2 | Enable the **New screen** command.<br>**Result:** A new screen with a generic name, appears in the tree directory |

## How to Modify the Name of a Screen

The Table Below Shows the Procedure for Modifying a Screen Name.

| Step | Action |
|---|---|
| 1 | In the project browser, right click on the screen to be modified.<br>**Result:** The relevant contextual menu appears |
| 2 | Enable the **Screen properties** command.<br>**Result:** The screen properties dialog box appears |
| 3 | Modify the screen name in the **Name** field and confirm with **OK**.<br>**Result:** The screen appears with a new name, in the tree directory |

## How to Delete a Screen

The Table Below Shows the Procedure for Deleting a Screen.

| Step | Action |
|---|---|
| 1 | In the project browser, right click on the screen to be deleted.<br>**Result:** The relevant contextual menu appears |
| 2 | Activate the **Delete** command.<br>**Result:** The screen is deleted from the directory tree |

## How to Move a Screen

The Table Below Shows the Procedure for Moving a Screen in the Tree Directory (For Example, Change Families).

| Step | Action |
|---|---|
| 1 | Right click on the screen to be moved, from the project browser.<br>**Result**: The relevant contextual menu appears |
| 2 | Enable the **Screen properties** command.<br>**Result:** The screen properties dialog box appears |
| 3 | Define in the **Location: Family** field, the name of the new family under which you want to place the screen. Click **OK** to confirm.<br>**Result:** The screen is placed in its new position To position it in the root (**Operator Screen** directory) leave the **Location: Family** field empty. |

**NOTE:** The screen can also be moved with the Drag-Drop function

## Operation of Screen Movement

When you move a screen from a family to the root, if:

- the family is associated with a functional module, the screen loses this link.

When you move a screen from a family to another **or** from the root to a family, if:

- the destination family screen is associated with a functional module, the screen will be also.
- the destination family screen is not associated with a functional module, the screen will not be associated with any functional module.

## How to Attach a Screen to a Functional Module

The Following Table Summarizes the Processes Used to Attach a Screen to a Functional Module.

| Step | Action |
|---|---|
| 1 | Display the structural view and the functional view. |
| 2 | In the project browser of the structural view, select the screen to be associated with a functional module. |
| 3 | Hold down the left mouse button to maintain the selection. |
| 4 | Drag and Drop the screen in the destination functional module of the functional view. The authorized insertion point of the screen in the destination functional module, is represented by a line.<br>**Result:**The screen is now associated with the chosen functional module |

### How to Detach a Screen from a Functional Module

The Following Table Summarizes the Processes Used to Detach a Screen from a Functional Module.

| Step | Action |
|---|---|
| 1 | In the functional view, right click on the screen to be detached from its functional module.<br>**Result:** The relevant contextual menu appears |
| 2 | Activate the **Detach** command.<br>**Result:**The screen loses its link with this functional module It can be associated with another functional module. |

### How to Delete All Screens and All Families

The Table Below Shows the Procedure for Deleting all the Screens and Families of a Project.

| Step | Action |
|---|---|
| 1 | In the structural view project browser, select the **Operator Screen** directory. |
| 2 | Right click on the **Operator Screen** directory.<br>**Result**: The relevant contextual menu appears |
| 3 | Activate the **Delete all** command.<br>**Result:**: A dialog box appears |
| 4 | Click **OK** to confirm your choice of deleting all screens and all families.<br>**Result:**: All screens and all families are deleted from the project |

### How to Export Screens

To find out the procedure for exporting operator screens *(see page 1747)*.

### How to Import Screens

To find out the procedure for importing operator screens *(see page 1749)*.

# Screen Family Editor Functions

## At a Glance

All Functions Can Be Accessed via the **Edit** Menu or by Right Clicking on the Browser Window in the Project Browser.

## How to Create a Screen Family

The Following Table Summarizes the Processes Used to Create a Screen Family.

| Step | Action |
|---|---|
| 1 | Right click the **Operator Screens** folder contained in the project browser tree directory.<br>**Result:** The relevant contextual menu appears |
| 2 | Enable the command **New family**.<br>**Result:**: A new family with a generic name appears in the directory tree. |

## How to Modify the Name of a Screen Family

The Table Below Shows the Procedures for Modifying the Name of a Screen Family.

| Step | Action |
|---|---|
| 1 | Right click on the family to be modified, from the project browser. |
| 2 | Enable the **Family properties** command.<br>**Result:**: The screen family dialog box appears |
| 3 | Modify the name of the family in the **Name** field, and click **OK** to confirm.<br>**Result:**: The screen family appears under a new name in the directory tree |

## How to Delete a Screen Family

The Following Table Summarizes the Processes Used to Delete a Screen Family.

| Step | Action |
|---|---|
| 1 | Right click on the family to be deleted, from the project browser. |
| 2 | Activate the **Delete** command.<br>**Result:**: The screen family is deleted from the directory tree |

### How to Move a Screen Family

The Following Table Summarizes the Processes Used to Move a Screen Family.

| Step | Action |
|---|---|
| 1 | Select the screen family to move in the project browser. |
| 2 | Hold down the left mouse button to maintain the selection. |
| 3 | Move the screen family to the desired position in the project browser. |
| 4 | Release the mouse button.<br>**Result:**: The screen family is moved to chosen place |

### How to Attach a Screen Family to a Functional Module

The Following Table Summarizes the Processes Used to Attach a Screen Family to a Functional Module.

| Step | Action |
|---|---|
| 1 | Display the structural view and the functional view. |
| 2 | In the project browser of the structural view, select the screen family to be associated with a functional module. |
| 3 | Hold down the left mouse button to maintain the selection. |
| 4 | Drag and Drop the screen family to the destination functional module of the functional view. The authorized insertion point of the screen family in the functional module, is represented by a line.<br>**Result:**: The screen family is now associated with the chosen functional module |

### How to Detach a Screen Family from a Functional Module

The following Table Summarizes the Processes Used to Detach a Screen Family from a Functional Module.

| Step | Action |
|---|---|
| 1 | In the functional view, right click on the screen family to be detached from its functional module.<br>**Result:** The relevant contextual menu appears |
| 2 | Activate the **Detach** command.<br>**Result:**: The screen family no longer has a link with a functional module It can be associated with another functional module. |

### How to Export a Screen Family

The export of a screen family takes place in the same way as screen export *(see page 1747)*.

# Operator screen properties

## At a Glance

When a screen is selected, the dialog box that is accessed by clicking **Edit** → **Screen properties** can be used to:

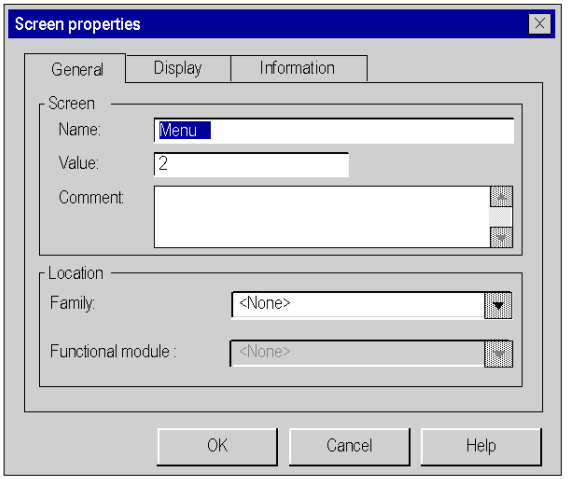
- Modify the screen label
- Modify the value associated with the screen
- Modify the comment associated with a screen
- Modify the screen position
- Modify the screen type
- Modify the color of the screen
- Modify the size of the screen
- Query its creation and modification dates
- Query the number of objects in the screen
- Query the number of variables used in the screen

The dialog box includes three tabs:

- **General**
- **Display**
- **Information**

## Screen properties dialog box

The screen properties dialog box is as follows:



### Tab: General

The following table describes the different functions offered by the **General** tab of the screen properties dialog box.

| Field | Description |
|---|---|
| **Name** | Screen name. It can be modified and can contain a maximum of 200 characters. |
| **Value** | Screen identification number You can modify this number which is used when you wish to associate a screen browse *(see page 1632)* button to the screen. This number can be used when the PLC *(see page 563)* controls the screen display in connected mode. |
| **Comment** | Screen comment |
| **Family** | Name of the family to which the screen belongs. The button to the right of the field is used to access the list of existing families. |
| **Functional Module** | Name of the functional module to which the screen is linked. The button to the right of the field is used to access the list of existing the functional modules. |

### Tab: Display

The following table describes the different functions offered by the **Display** tab of the screen properties dialog box.

| Field | Description |
|---|---|
| **Used as Pattern** | When this box is checked, the screen may be used as a pattern for any project screen. A pattern screen cannot contain animated objects. |
| **Uses pattern** | When this box is checked, the background of this screen uses the background pattern that you have defined. The right button can be used to select the model screen. |
| **Default color** | When this box is checked, the screen background is the default color used by Windows software for windows. |
| **Custom color** | When this box is checked, the screen background is the color you defined (color of the field located on right side of the check box). The button allows access to the screen color settings. |
| **Size** | Screen sizes. It can be changed and its default value is 1280 pixels by 1024 pixels. |

## Tab: Information

The following table describes the different functions offered by the **Information** tab of the property dialog box of a screen.

| Field | Description |
|---|---|
| **Information** | This information cannot be modified; it indicates:<br>● The screen creation date<br>● The last screen modification date<br>● The number of objects contained in the screen<br>● The number of variables used in the screen |

# Properties of Operator Screen Families

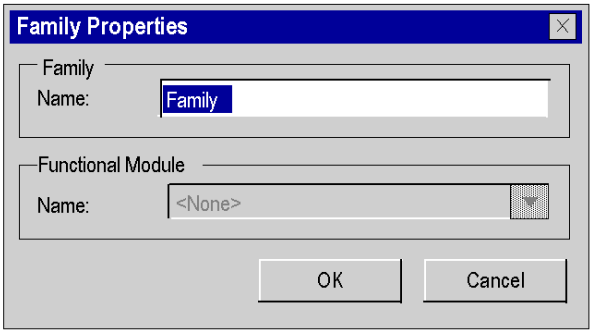
## At a Glance

When a screen family is selected, the dialog box accessible via the **Edit** → **Family Properties** command allows you to:

- modify the name of a screen family
- associate the family with a functional module.

## Properties dialog box of a family

The properties dialog box of a family is as follows.




## Description

The following table describes the different functions offered by the properties dialog box of a family.

| Field | Description |
|---|---|
| **Family** | Name of the family to which the screen belongs. It can be modified and can contain up to 200 characters. |
| **Functional Module** | Name of the functional module with which the family is associated. The button to the right of the field is used to access the list of existing functional modules. |

**NOTE:** A functional module cannot be created from this dialog box.

# Section 48.2
## Creating an object

### Subject of this section

This section describes how to create or insert an object.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction of a graphic screen object | 1603 |
| How to create standard objects | 1604 |
| How to insert an image in a screen | 1606 |
| How to create driving objects | 1608 |
| Operation of the driving objects | 1610 |

# Introduction of a graphic screen object

## General

4 types of objects can be created in a graphic screen :

- standard objects: line, rectangle, ellipse, curve, polyline, text,
- images: bitmap files with the **BMP** or **JPG**extension,
- the driving (or command) objects: button, check box, input field, counter, cursor, explicit exchange object, screen browser button,
- compound objects: group of objects of previous 3 types, created by the user or from the object library.

All the objects used to create a screen can be static or animated.

# How to create standard objects

## At a Glance

Standard objects are the basic elements of graphic screens.

They are the simple objects which allow the composition of complex objects.

There are two types of standard objects:

- Static objects
- Dynamic or animated objects

Static objects do not have variables associated with them. Their graphic representation is fixed.

Dynamic objects possess one associated variable, which allows their display to be modified.

## How to create a standard object

The create standard object procedure is identical regardless of the object, with the exception of polygons.

| Step | Action |
|---|---|
| 1 | Select the type of object that you wish to create from the tool palette. |
| 2 | Move the mouse cursor to the position where you want to place the top left corner of your object. |
| 3 | Press down the left mouse button. |
| 4 | Hold the button down and move the mouse cursor to the position where you want to place the bottom right corner of the object. |
| 5 | Release the button.<br>**Result**: The object is created when you release the mouse button |
| 6 | Right click on the object.<br>**Result**: The contextual menu appears |
| 7 | Enable the **Properties** command.<br>**Result**: A contextual dialog box appears This box depends on the type of object and allows the attributes *(see page 1617)* to be fixed. |

## How to create a polygon object

The procedure for creating a polygon object is as follows.

| Step | Action |
|---|---|
| 1 | Select the type of polygon object that you require from the tool palette. |
| 2 | Move the mouse cursor to the position where you want to place the first point of your object. |
| 3 | Press the left mouse button then release.<br>**Result**: The object's start point is created |
| 4 | Move the mouse cursor to the position where you want to end the first line. |
| 5 | Press the left mouse button then release.<br>**Result**: You have just created the first line of your object |
| 6 | Repeat steps 4 and 5 to create new lines to form your polygon object.<br>**Result**: Segments of consecutive straight lines are created |
| 7 | Finish by left clicking on the end point of the object.<br>**Result**: The object is created |
| 8 | Right click on the object.<br>**Result**: The contextual menu appears |
| 9 | Enable the **Properties** command.<br>**Result**: A contextual dialog box appears This box depends on the type of object and allows the attributes *(see page 1617)* to be fixed. |

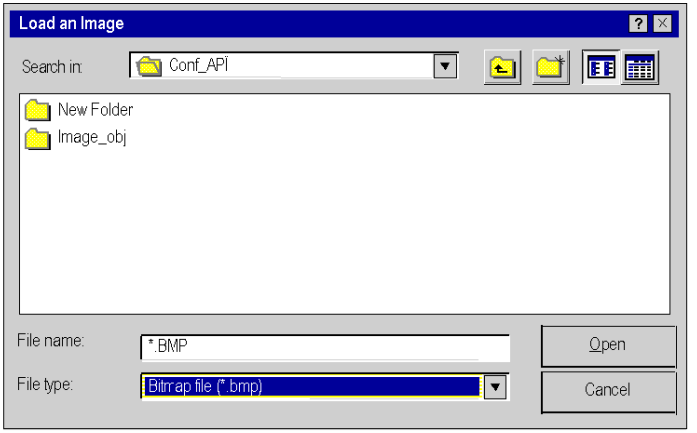
# How to insert an image in a screen

## At a Glance

An operator screen can contain one or several bmp, jpg or jpeg format images.

## How to insert an image object in a screen

Follow the steps below to insert an image in a screen.

| Step | Action |
|---|---|
| 1 | Select the image icon from the tool palette.<br>**Result:** the selection window of the bitmap file to be inserted appears<br>  |
| 2 | Select the type of file to display (bmp, jpg or jpeg). |
| 3 | Select the file to be inserted and enable the **Open** command.<br>**Result:** the image is inserted in the screen It appears with selection handles. |

## Rules and precautions for use

Only images in bmp, jpg or jpeg formats are recognized.

Optimize loading time:

- Do not integrate too many images into one screen
- Limit the size of images
- Avoid modifying the image's proportions and size in the screen

Optimize display quality:

- Avoid modifying the image's proportions and size in the screen
- If a screen contains several different images, use 24 bit screen depth

If the image *(see page 1636)* is not copied in the project, it will not be exported, during a screen export *(see page 1747)*.

# How to create driving objects

## At a Glance

Driving (or command) objets allow the user to perform an action:

- Browse from one screen to another
- Modify the value of a variable
- Send a command to the PLCs task module

There are 7 types of these objects:

- Buttons
- Check boxes
- Input zones
- Counters
- Cursors
- Buttons for browsing between screens
- Objects for explicit exchanges

## How to create a driving object

The procedure for creating a driving object is as follows.

| Step | Action |
|---|---|
| 1 | Select the type of driving object that you wish to create from the tool palette. |
| 2 | Move the mouse cursor to the position where you want to place the top left corner of your object. |
| 3 | Press down the left mouse button. |
| 4 | Hold the button down and move the mouse cursor to the position where you want to place the bottom right corner of the object. |
| 5 | Release the button.<br>**Result:** The object is created when you release the mouse button |
| 6 | Enable the **Selection** tool from the tool palette. |
| 7 | Right click on the object.<br>**Result:** The contextual menu appears |
| 8 | Enable the **Properties** command.<br>**Result:** A contextual dialog box appears This box depends on the type of object and allows the attributes *(see page 1617)* to be fixed. |

### Type of data driven according to the driving object

The table indicates the types of variables that may be driven by the driving objects.

| Object | Bit | Byte | Word | Double word | Floating point |
|---|---|---|---|---|---|
| Button | X | X* | X* | X* | X* |
| Check box | X | - | - | - | - |
| Entry field | X | X | X | X | X |
| Counter | X | X | X | X | X |
| Cursor | X | X | X | X | X |
| | | | | | |
| Legend: | | | | | |
| * : Only in the case of unsynchronized buttons | | | | | |

**NOTE:** Driving objects are always placed in the foreground. To prevent command errors, we strongly advise against placing driving objects between them.

# Operation of the driving objects

## At a Glance

The piloting objects are enabled by clicking on the mouse (or via the keyboard). Depending on the fixed attribute, these objects act on their associated variables.

**NOTE:** The state of associated variables can be changed by the PLC application. The action of the driven object doesn't constitute a forcing of the variable.

Furthermore, so long as the input of a value has not been validated or that the driven object maintains control, the displayed value may differ from the real value of the variable contained in the PLC memory.

## How the buttons operate

Depending on its attributes, a button works as follows.

| When you click on the button, if the associated variable is... | and if | Then |
|---|---|---|
| a bit, | driving is unsynchronized, | the associated variable switches to 1 when the left mouse button is held down, and switches back to 0 when released. |
| digital, | driving is unsynchronized, | the associated variable takes the fixed threshold value in the button attributes. |
| a bit, | driving is synchronized, | the state of the associated variable changes. It moves to 0 if it was at 1, and vice versa. |

## How a check box operates

A check box works as follows.

| When the box | Then |
|---|---|
| is checked, | the associated bit is set to 1. |
| isn’t checked, | the associated bit is set to 0. |

### How an input zone operates

An input zone works as follows.

| When | and if | Then |
|---|---|---|
| the entry zone contains a value, | you press **Enter**, | the associated variable takes the value read in the entry zone. |
| the associated variable is modified by the PLC, | you did not enter a value in the entry zone, | the zone displays the value read in the PLC. |

### How a counter operates

A counter operates as follows.

| When | and if | Then |
|---|---|---|
| the counter is horizontal, | you click on the right arrow of the counter, | the associated variable is incremented by one unit. |
| the counter is horizontal, | you click on the left arrow of the counter, | the associated variable is decreased by one unit. |
| the counter is vertical, | you click on the top arrow of the counter, | the associated variable is incremented by one unit. |
| the counter is vertical, | you click on the bottom arrow of the counter, | the associated variable is decreased by one unit. |

**NOTE:** The increment and decrement value is multiplied by 10, 100 then 1000 when the arrow is held down.

### How a cursor operates

A cursor operates as follows.

| When the cursor | and if | Then |
|---|---|---|
| is horizontal, | it is moved to the right, | the associated variable is incremented proportionally to the move made and to the minimum and maximum thresholds defined for the variable. |
| is horizontal, | it is moved to the left, | the associated variable is decremented proportionally to the move made and to the minimum and maximum thresholds defined for the variable. |
| is vertical, | it is moved upwards, | the associated variable is incremented proportionally to the move made and to the minimum and maximum thresholds defined for the variable. |
| is vertical, | it is moved downwards, | the associated variable is decremented proportionally to the move made and to the minimum and maximum thresholds defined for the variable. |

### How an inter-screen browser button operates

An inter-screen browser button operates as follows.

| When the button | Then |
|---|---|
| is enabled by a click, | the screen with the associated number is displayed. |

**NOTE:** In the case of Multi-Instance *(see page 550)* mode, clicking on the browser button opens a new window with the specified screen. If this is not the case, the new screen opens in the current window.

### How an explicit exchange button operates

An explicit exchange button operates as follows.

| When the button | Then |
|---|---|
| is enabled by a click, | the associated explicit exchange is launched. The memory zone of the PLC and that of the module are updated according to the configured *(see page 1638)* parameters. |

# Section 48.3
# Default Properties screen

## Default Properties screen

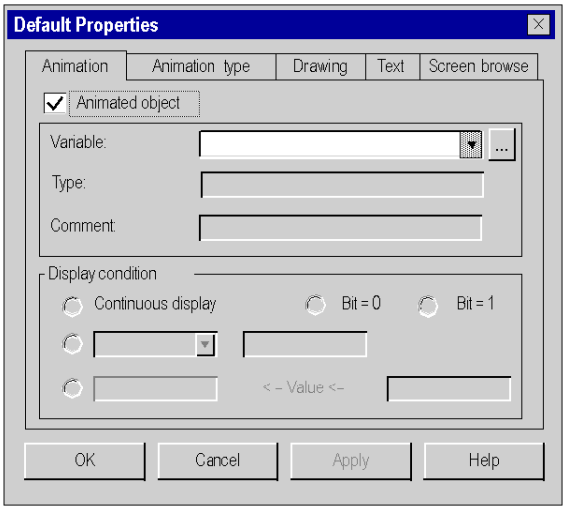
### At a Glance

The Default properties screen allows to set common attributes for all objects you will place in the operator screen.

You can access this screen by right-clicking on the operator screen and select properties in the menu. (No object must be selected)

### Default properties screen

The following screen shows the different tabs of the default properties screen. (In this example animation tab is visible)



## Description

To see the description of all the tabs refer to object properties tab screens.

- Animation tab *(see page 1623)*
- Animation type tab *(see page 1625)*
- Drawing tab *(see page 1618)*
- Text tab *(see page 1621)*
- Screen browse tab *(see page 1632)*

## Behaviour of the Default Properties screen

When you modificate attributes in the Default Properties screen, the new attributes are taken into account only for the objects you will place in the operator screen after the modifications. If there are already objects in the operator screens, the modifications will not impact them. In this case you have to modify the attributes for each object (or compound of objects).

# Section 48.4
## Modification of an object’s attributes

### Subject of this section

This section describes the attributes of objects, how to access these attributes and the associated information.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| How to access an object’s attributes | 1617 |
| Design tab | 1618 |
| Extended colors | 1620 |
| Text tab | 1621 |
| Animation Tab | 1623 |
| Animation type tab | 1625 |
| Trend diagrams | 1628 |
| Bar chart | 1630 |
| Screen browse tab | 1632 |
| Control tab | 1634 |
| Image Tab | 1636 |
| Explicit Exchanges Tab | 1638 |
| Information on the objects | 1640 |
| The data selection tool | 1642 |

# How to access an object's attributes

## At a Glance

Each object created in a screen has attributes which depend on its type. These attributes are accessed via the tabs below.

- Animation *(see page 1623)*
- Type of animation *(see page 1625)*
- Design *(see page 1618)*
- Text *(see page 1621)*
- Image *(see page 1636)*
- Driving *(see page 1634)*
- Screen browsing *(see page 1632)*
- Explicit exchanges *(see page 1638)*

Access to these attributes is the same for all created objects.

## How to access an object's attributes

The table below indicates how to access the settings of object attributes.

| Step | Action |
|---|---|
| 1 | Select the object whose attributes you wish to modify. |
| 2 | Click on the object with the right mouse button.<br>**Result:** The contextual menu appears |
| 3 | Enable the **Properties** command.<br>**Result:** A dialog box displays the attributes linked to the kind of object chosen. |

# Design tab

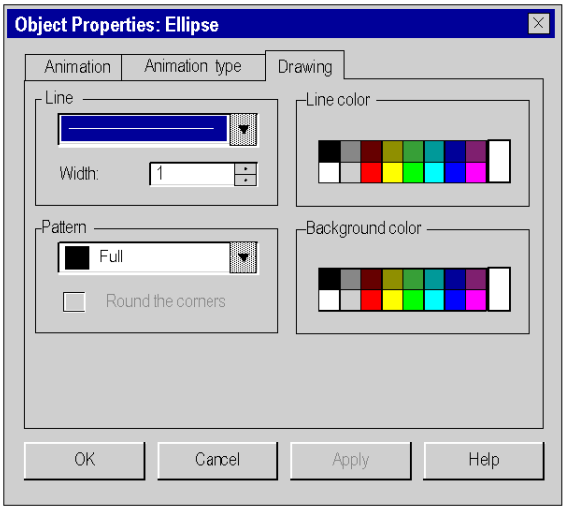
## At a Glance

This tab is used to set the design attributes of a graphic object:

- Modification of the object's contour line
- Modification of the object's pattern and color

## Design attributes setting screen

The screen below is used to set the design attributes of a rectangle type graphic object.



## Description

The table below describes the settings that you may define.

| Field | Description |
|---|---|
| **Line** | Contour line of the graphic object:<br>● Single line<br>● Single or complex dotted line<br>● No contour line |
| **Thickness** | Thickness of the object's contour line. It is defined in pixels and may range from 1 to 10 pixels maximum. |
| **Outline color** | Color of contour line. The color palette contains 16 predefined colors and provides access to the extended colors *(see page 1620)* parameter screen. |
| **Pattern** | The pattern inside the object. This pattern is defined by a frame and a background color. The available frames are as follows:<br>● None (the background is transparent)<br>● Fill (filled with the selected background color)<br>● Horizontal frame<br>● Vertical frame<br>● Diagonal frame to the top<br>● Diagonal frame to the bottom<br>● Grid<br>● Lattice |
| **Round the corners** | When this box is checked, the corners of rectangle type objects are rounded. |
| **Background color** | Color of the object's content (background). The colors palette contains 16 predefined colors and provides access to the extended colors *(see page 1620)* parameter screen. |

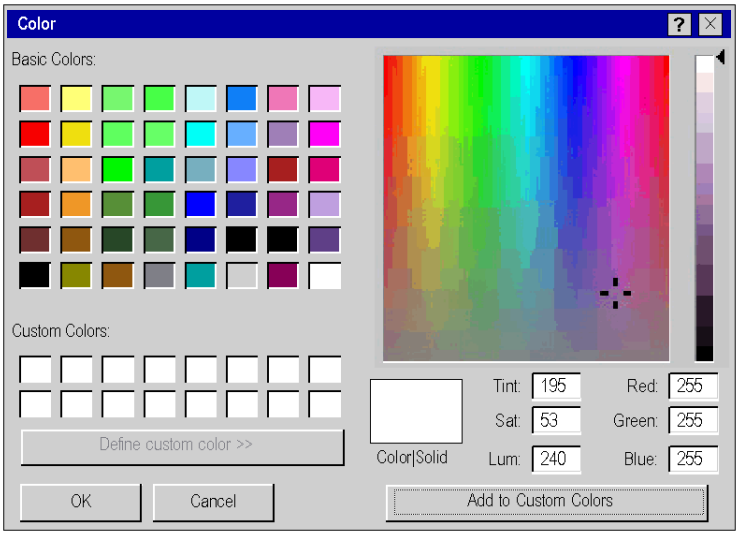
# Extended colors

## At a Glance

This dialog box proposes 48 basic colors and allows you to create personalized (RGB) colors enhanced with tint, saturation and brightness.

## Color parameter screen

The following screen allows you to create or modify the extended colors.




## Creating a personalized color

This table describes how to create a new color, that can be used for the graphic objects.

| Step | Action |
|---|---|
| 1 | Position the cross hair for selecting color, tint and saturation to find the required shade. |
| 2 | Adjust (up or down) the light and color selection cursor to refine your choice. |
| 3 | Select a box from **Custom Color**. |
| 4 | Click on the **Add to custom colors** button to create a new color. |

# Text tab

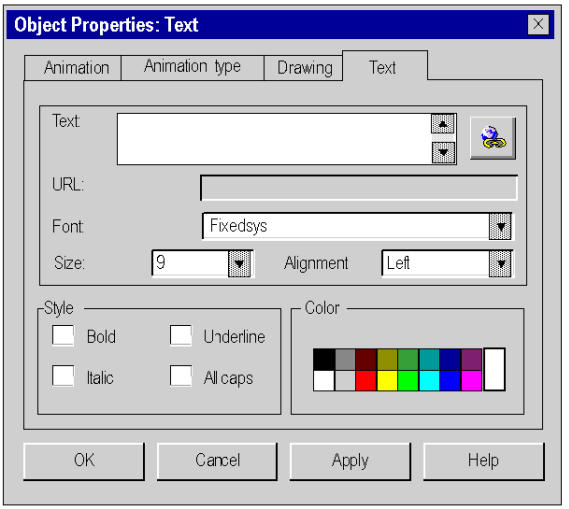
## At a Glance

This tab is used to set the text attributes of a graphic object:

- text input,
- associate a hyperlink,
- select the character font,
- select font size,
- set text alignment,
- select text style,
- select text color.

## Text attributes setting screen

The screen below is used to set the text attributes of a graphic object.



## Description

The table below describes the settings that you may define.

| Field | Description |
|---|---|
| **Text** | Text of the object. 32635 maximum characters entered on several lines, separated by a carriage return. If the related variable is of the real type, users may define a display format as #XX.XXX# (in this case a 5 figure display with three decimals). |
| **button**  | Allows access to the dialog box to define the hyperlink. This hyperlink *(see page 1817)* will be associated with the text attribute of the graphic object. |
| **Url** | Static display of the hyperlink defined by the  button. |
| **Font** | Character font. The fonts available for selection are those installed in Windows. |
| **Size** | Font size. |
| **Alignment** | Text alignment: align left, right or centered in relation to the display area. |
| **Style** | Text style: bold, underlined, italics or capitals. |
| **Color** | Text color. The colors palette contains 16 predefined colors and provides access to the extended colors *(see page 1620)* parameter screen. |

# Animation Tab

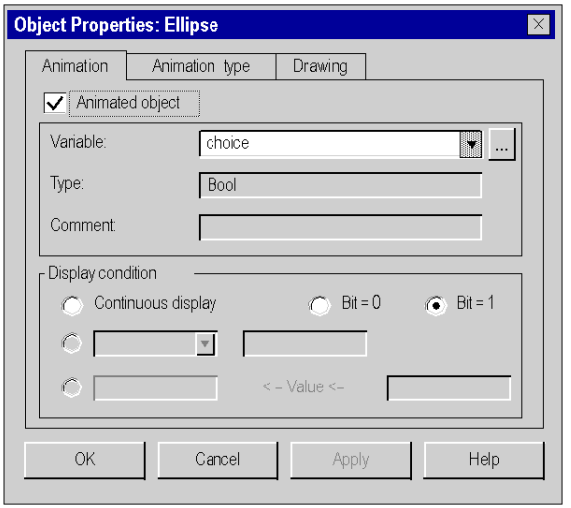
## At a Glance

All the graphic objects (simple or compound) that you create in a screen can be animated. This tab allows users to set the animation attributes of a graphic object.

- to decide to animate an object,
- to choose the animation variable,
- to choose the display conditions.

## Animation attributes setting screen

The following screen is used to set the animation attributes of a graphic object.



## Description

The table below describes the settings that you may define.

| Field | Description |
|---|---|
| **Animated object** | The object is animated if this box is checked. |
| **button**  | Calls the variable selection tool to select the variable to associate with the object. |
| **Variable** | The animation variable linked to the object. Possibilities:<br>● simple variables,<br>● the public variables of a function block,<br>● the parameters (input, output, input/out) of a function block. |
| **Type** | Type of animation variable linked to the object. The type must be EDT (Bool, Ebool, Int, Dint, Uint, Udint, Real, Time...). It is not possible to use structures or tables. However, it is possible to use items from tables or structures if they are of type EDT. |
| **Comment** | Comment of the animation variable linked to the object. |
| **Display condition** | The object's display condition:<br>● Permanent: the graphic object is always present on the screen,<br>● Bit = 0: the graphic object is displayed if the associated variable (bit) is in state 0,<br>● Bit = 1: the graphic object is displayed if the associated variable (bit) is in state 1,<br>● <, >, =, <>: the graphic object is displayed if the defined condition is true.<br>● < = value < =: the graphic object is displayed if the defined condition is true. |

# Animation type tab

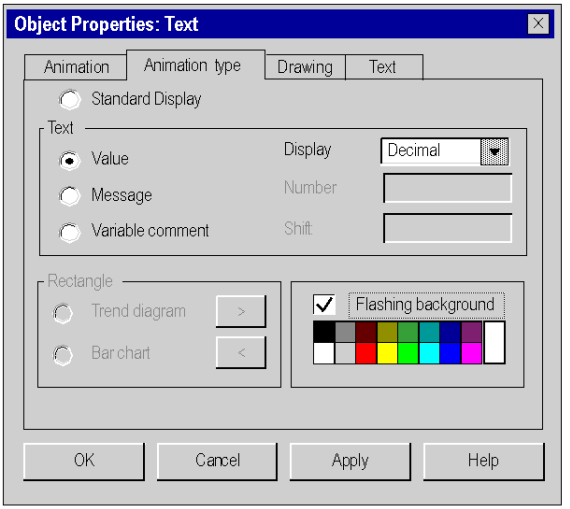
## At a Glance

All the graphic objects (simple or compound) that you create in a screen can be animated. This tab is used to set the animation type which depends on the nature of the graphic object to be animated:

- to select the object's display mode,
- to select the display mode of the text linked to the object,
- to select flashing option,
- to select the trend diagram mode,
- to choose bar chart mode.

## Animation type attributes setting screen

The following below is used to set the animation type attributes of a graphic object.



## Description

The table below describes the settings that you may define.

| Field | Description |
|---|---|
| **Standard Display** | The object is displayed with its attributes if this button is selected. |
| **Value** | The value taken by the variable is displayed according to the display mode chosen.<br>In offline mode, even if this animation style is selected, the text defined in the **Text** field of the object is displayed on the screen with the selected font and size. |
| **Display** | Choice of display mode:<br>● Decimal,<br>● Hexadecimal (value preceded by 16#),<br>● Binary (value preceded by 2#),<br>● ASCII.<br>When the length of the object's definition zone is too small to display the value, the field is filled with **#** characters. The font used is the one defined in the Text tab *(see page 1621)*.<br>If the animation variable is of the **String** type (string of characters), the **Display** field cannot be accessed because viewing is automatically in ASCII. |
| **Message** | You must select this button to display a message configured in the Operator screens messages editor *(see page 1656)*. The **Number** field allows you to enter either a message number, or the term Val. In the last case, the displayed message will be the one whose value is contained in the variable associated with the object. The **Shift** field allows you to enter a value that will be added to the one contained in the variable. |
| **Variable comment** | If this button is selected and if the text animation condition is met, the comment associated with the variable is displayed.<br>**Note**: This function allows, among other things, to avoid a double entry. Enter the comment once in the variables editor. This will be used as a message linked to the variable. |
| **Trend diagram** | You may use rectangle type objects such as trend diagrams *(see page 1628)*. Selecting this box will give you access to the trend diagram's parametering box. |
| **Bar chart** | You may use rectangle type objects such as bar charts *(see page 1630)*. Selecting this box will give you access to the bar chart's parametering box. |
| **Flashing background** | When this box is selected and if the display condition is met, the background of the graphic object flashes according to the user-specified flashing color. |

## Properties

The table below shows the types of animation that may be associated with the driving objects.

| Object | Standard | Value | Message | Trend diagram | Bar chart | Flashing |
|---|---|---|---|---|---|---|
| Line | X | - | - | - | - | X |
| Rectangle | X | - | - | X | X | X |
| Ellipse | X | - | - | - | - | X |
| Curve | X | - | - | - | - | X |
| Polygon | X | - | - | - | - | X |
| Text | X | X | X | - | - | X |
| Compound object | X | - | - | - | - | - |
| Image | X | - | - | - | - | - |
| Control | X | - | - | - | - | - |
| | | | | | | |
| Legend: | | | | | | |
| X: Yes<br>- : No | | | | | | |

# Trend diagrams

## At a Glance

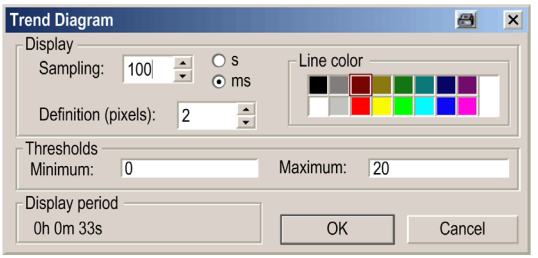
A trend diagram allows the graphic representation of a variable status, with the representative curve moving from right to left.

You are advised to limit the number of trend diagrams on a same page because their display decreases the system's performance.

Only rectangular type objects can be used as trend diagrams.

## Trend diagram parameter screen

The screen below allows to parameter the animated rectangles of the trend diagram format.




## Description

The table below describes the settings that you may define.

| Attribute | Description |
|---|---|
| **Sampling** | Period in which a new value is added at the end of the diagram and the diagram is refreshed.<br>Default value: 1 second.<br>Values between 1 millisecond and 3600 second are allowed. Sampling periods below 100 ms produce a lot of windows messages and generate a high CPU load. Therefore only few diagrams with such small periods should be animated at the same time and long lasting actions like **Build Changes** should be avoided while diagrams are animated (see also the **NOTE** at the end of the table). |
| **Definition** | Length of value display. Each new value is represented by a right segment whose length is fixed by the definition.<br>The definition is expressed in pixels between 1 and 20.<br>The value by default is 2. |
| **Line color** | Color of the timing diagram's right segments<br>The colors palette contains 16 predefined colors and provides access to the extended colors *(see page 1620)* parameter screen. |

| Attribute | Description |
|---|---|
| **Threshold: minimum and maximum** | Thresholds between which the value representation develops. For binary values, the thresholds must be 0 and 1. |
| **Display period** | This value is calculated. It depends on the size of the rectangle, the sampling value and the definition.<br>The procedure is as follows:<br>Period = Rectangle size / Definition x Sampling. |

**NOTE:** Internally, for every diagram, a timer is started with the sampling period defined by the user. When the timer fires, the last value of the corresponding variable transmitted from the PLC is read and added at the end of the diagram. Such timers may not always fire exactly in equidistant periods (e.g. they may be delayed when a lot of tasks run in parallel on the PC). Therefore the displayed values may not represent the exact values in the PLC at the end of each sampling period.

# Bar chart

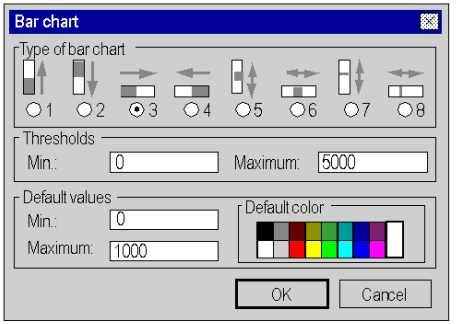
## At a Glance

Bar charts allow the graphic representation, in rectangular format, of the status of a variable.

Only rectangular type objects can be used as bar charts.

## Bar chart parameter screen

The screen below allows parameterizing of animated rectangles in bar chart format.




## Description

The table below describes the settings that you may define.

| Attribute | Description |
|---|---|
| **Type of bar chart** | **1.** The bar chart moves from bottom to top.<br>**2.** The bar chart moves from top to bottom.<br>**3.** The bar chart moves from left to right.<br>**4.** The bar chart moves from right to left.<br>**5.** The bar chart moves symmetrically in relation to a horizontal axis corresponding to the average of the thresholds.<br>**6.** The bar chart moves symmetrically in relation to a vertical axis corresponding to the average of the thresholds.<br>**7.** The bar charts grow vertically, by symmetrically filling the gap between the value of the variable and the average of the thresholds.<br>**8.** The bar charts grow horizontally, by symmetrically filling the gap between the value of the variable and the average of the thresholds. |
| **Thresholds** | Values indicating the range of status displayed. When the associated variable value is equal to the minimum threshold value, the bar chart is empty. When this value is equal to the maximum limit value, the bar chart is full (the rectangle's color).<br>By default these values are set to 0 (minimum) and 1000 (maximum). |

| Attribute | Description |
|---|---|
| **Default values** | Values that indicate the thresholds from which the bar chart is displayed with the default color.<br>The bar chart uses the default color when the associated variable value is:<br>● higher than or equal to the maximum value<br>● lower than or equal to the minimum value<br>The colors palette contains 16 predefined colors and provides access to the extended colors *(see page 1620)* parameter screen. |

# Screen browse tab

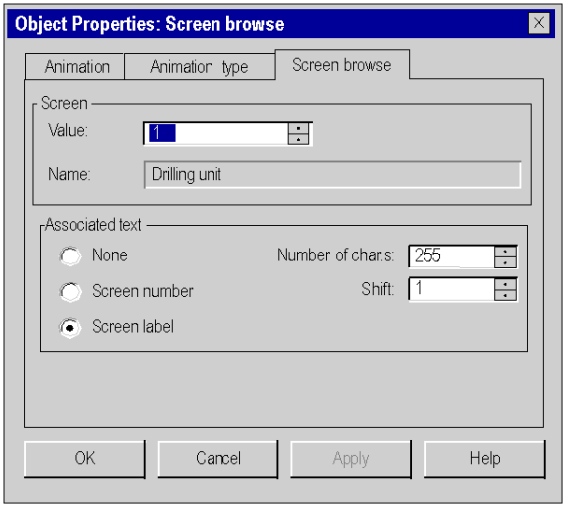
## At a Glance

The screen browse buttons allow users to browse between screens (move from one screen to another). The parametering of a button's attributes links the button to the screen. The button's browser attributes allow users to:

- choose the screen to be displayed,
- define the text displayed in the button,

## Screen browse attributes setting screen

The screen below allows users to set the animation attributes of a screen browse button.



## Description

The table below describes the different parameters that you may define.

| Field | Description |
|---|---|
| **Value** | This field allows entry of the screen that is displayed when you click on the button. Two cases are possible:<br>● the value entered is a number that indicates the screen number to display,<br>● you have entered the term **VAL**. The screen displayed will be the one whose number is contained in the animation variable linked to the button object (See *Animation Tab, page 1623*). |
| **Name** | This field indicates the label of the screen you have defined by its number in the **Value** field. If you have defined the screen by the term **VAL**, this field displays the characters **## ? ##**. |
| **None** | If this box is checked, no text is associated with this button. |
| **Screen number** | If this box is checked, the screen number is displayed on the button. |
| **Screen label** | If this box is checked, the screen label is displayed on the button. |
| **Number of char.s** | If the **Screen label** box is checked, the field allows you to define the number of characters to display in the button. |
| **Shift** | If the **Screen label** box is checked, the field indicates the position (shift) of the first character of the label to display in the button. |

**NOTE:** Each browser button has a logical number that links it to a screen. If this number is modified, the button will no longer point on the screen.

**NOTE:** The software does not take into account the deletion or import of screens.

# Control tab

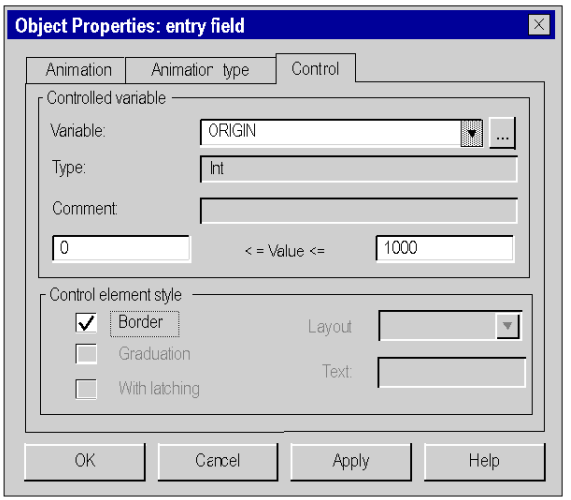
## At a Glance

This tab allows you to define the characteristics of a control object:

- to choose a control variable,
- to choose the minimum and maximum values,
- to determine the monitoring style.

## Control attributes setting screen

The screen below allows users to set the control attributes of a command graphic object.



## Description

The table below describes the settings that you may define.

| Field | Description |
|---|---|
| **Variable** | Variable controlled by the graphic object. |
| **button**  | Calls the variable selection tool to select the variable to associate with the object. |
| **Type** | Type of variable linked to the object. |
| **Comment** | Comment of the variable linked to the object. |
| **< = value <=** | The control object will not exceed these thresholds.<br>By default, the minimum value is 0 and the maximum value is 100.<br>If the variable to control is a bit, these fields are not accessible. |
| **Control element style** | Control style which depends on the type of control object. It allows you to modify the representation of the graphical object:<br>● **Border** displays a boarder around the object,<br>● **Graduation** displays color graduations,<br>● **With latching** denotes a button with latching,<br>● **Orientation** determines the orientation of the object on the screen,<br>● **Text** displays the text with the object. |

**NOTE:** If the graphic object is a button, and if the control variable is a word, a double word or a real word, the maximum threshold becomes a **dynamic threshold**. It is the value of this threshold that is sent when you click on the button

# Image Tab

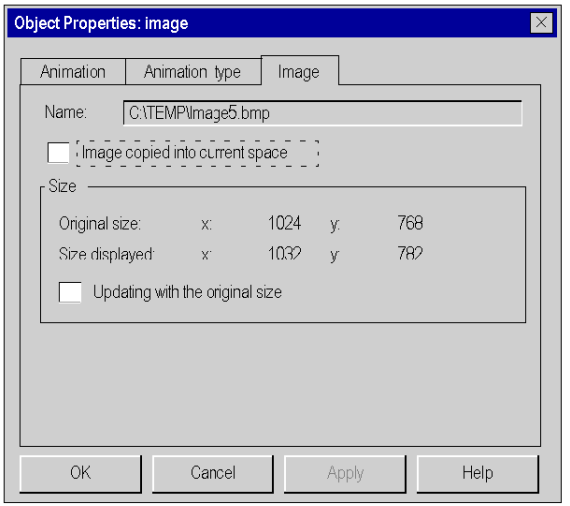
## At a Glance

This tab allows you to set the attributes of the .BMP, JPG or JPEG format images built into the operator screens:

- To choose the image to be displayed,
- To integrate the image into the project,
- To query the size of the image (real and visual).

## Image Attributes Setting Screen

The screen below allows you to set the attributes of a bitmap image built into an operator screen.



## Description

The table below describes the settings that you may define.

| Field | Description |
|---|---|
| **Name** | This field indicates the name and the access path to the file. |
| **Image copied into current space** | When this box is checked, the file is copied and compressed in the project file (.STU or STA for an archive). Loading is faster when using the screens and the image is exported at the same time as the screen during an export *(see page 1747)*.<br>If the image is already built into the project, this box does not appear. |
| **Size** | This information, accessible in read only gives information on the original size and the displayed size of the image. Its is used to know if the image has been modified. |
| **Updating with the original size** | When this box is checked, the image changes to its original size.<br>This box does not appear if the original size and the displayed size of the image are identical. |

# Explicit Exchanges Tab

## At a Glance

Some Bits and Words Are Not Implicitly Exchanged (at Each PLC Cycle) Between the Modules and the PLC Processor. These bits and words are grouped into 3 families: information on status (STATUS), command (CMD) and parametering (PARAM) and are explicitly exchanged.

Explicit exchanges are normally made by program, but the user can also initiate such an exchange by clicking on a button, located in an operator screen.

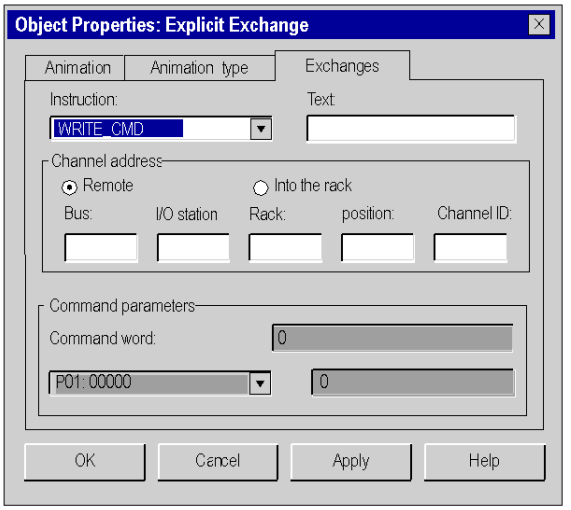
The explicit **exchange** tab allows you to define the exchange characteristics:

- to select an explicit exchange instruction,
- to select the text of the button,
- to select the address of the channel to which the exchange will be applied,
- To display the parameters associated with the instruction.

**NOTE:** Only the exchanges of objects associated with a **channel** are authorized.

## Explicit Exchanges Attributes Setting Screen

The screen below allows you to set the attributes of an explicit exchange object.



## Description

The table below describes the settings that you may define.

| Field | Description |
|---|---|
| **Instruction** | Explicit exchange instruction:<br>● READ_STS: reading of channel status words contained in the module and updating the data in the PLC.<br>● WRITE_PARAM: writing of channel parametering words, contained in the module.<br>● READ_PARAM: reading of channel parametering words, contained in the module.<br>● SAVE_PARAM: backup in a specific memory space of channel parametering words, contained in the module.<br>● RESTORE_PARAM: restoration of values saved by SAVE_PARAM.<br>● WRITE_CMD: implements the execution of a command on a channel. |
| **Text** | Text that appears on the button. |
| **Channel address** | **Remote**: this box is checked if the module to be interrogated is on a remote station.<br>**In rack**: this box is checked if the module to be interrogated is on the rack of the local station.<br>Address of the module channel. This address has the form:<br>● \bus.station\rack.location.channel:<br>❍ bus: bus number (shaded if local station),<br>❍ station: number of the device connection point (shaded if local station),<br>❍ rack: station rack number,<br>❍ position: position of the module on the rack,<br>❍ channel: channel number. |

**NOTE:** For Write_Param and for Write_Cmd we have to modified the associated %MW by animation table or another one (by program, by 'Entry field' through operator screen, ...) before sending the instruction (Write_Param or Write_Cmd) through the Operator screen.

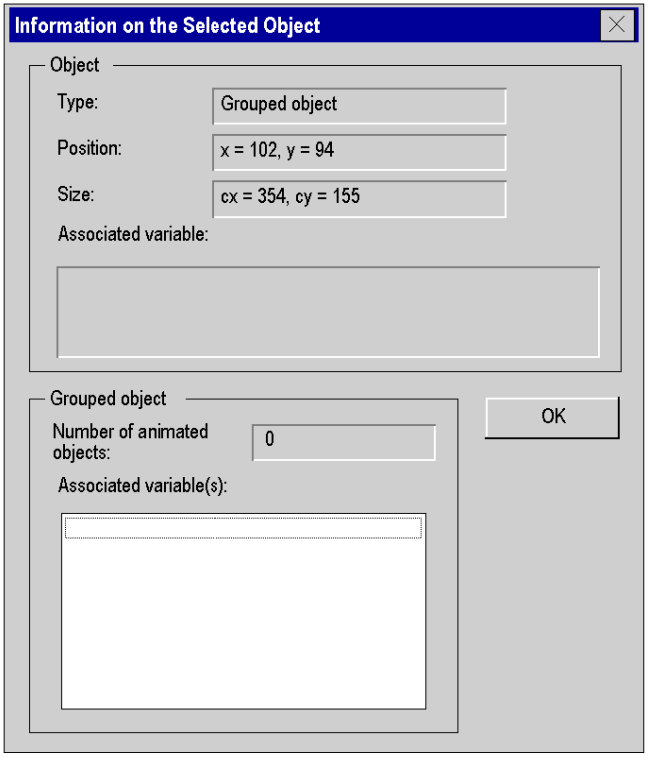
# Information on the objects

## At a Glance

The information screen on the objects, accessible via the **Edit** → **Information** command allows the information on each of the objects contained in a screen to be displayed. This screen is particularly useful for compound objects.

## Information screen on an object

The screen below provides information on a graphic object.



## Description

The following table describes the information offered by this screen.

| Information | Description |
|---|---|
| **Type** | Type of object. ellipse, button, cursor, object compound, etc. |
| **Position** | Coordinates of the object's top left point in pixels. |
| **Size** | Size of the object in pixels. |
| **Associated variable** | Variable linked to a simple object or controlled by a simple object. In the case of a compound object, this field is left blank. The information is placed in the **Grouped object** zone. |
| **Grouped object** | Number of simple animated objects contained in a compound object and the variables associated with (or controlled by) this compound object. In the case of a simple-type object, this field is not displayed. |

# The data selection tool

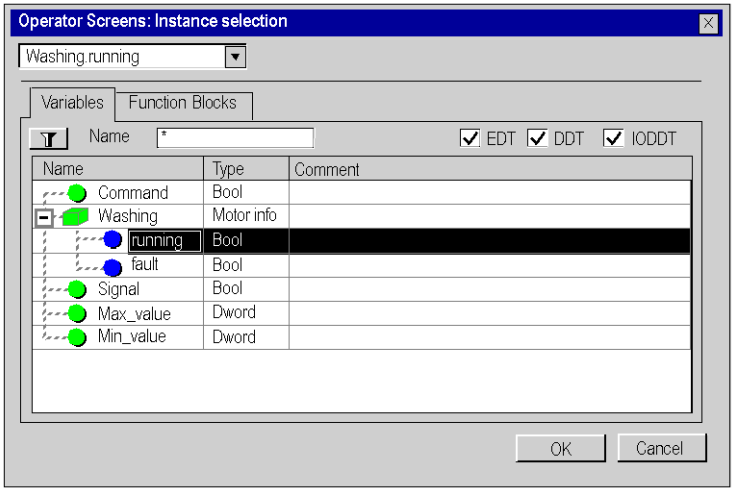
## At a Glance

All the graphic objects (simples or compound) that you create in a screen can be animated *(see page 1623)*. The variables selection tool allows you to associate a variable to animate the object. Variables that can animate objects include:

- simple variables, *(see Unity Pro, Program Languages and Structure, Reference Manual )*
- the public variables *(see Unity Pro, Program Languages and Structure, Reference Manual )*of a function block,
- the parameters (input, output, input/out) *(see Unity Pro, Program Languages and Structure, Reference Manual )*of a function block.

## Procedure

The table below presents the procedure for using a variable to animate an object.

| Step | Action |
|---|---|
| 1 | Select to object to parameter in the operator screen. |
| 2 | Right click in the contextual menu to select the **Properties** command.<br>**Result:** A dialog box displays the attributes linked to the kind of object chosen. |
| 3 | Select the animation tab. |
| 4 | Check the **Animated Object** box. |
| 5 | In the **Variable** field enter the name of the variable or use the  button to open the variables selection tool.<br>**Result:** The variables selection tool appears.<br>  |
| 6 | From the **Variables** tab or the **Function Blocks** tab select the variable to be used to animate the object. |
| 7 | Click **OK** to confirm. |

### Variables tab

The table below describes the elements of the **Variables** tab of the data selection tool.

| Field | Description |
|---|---|
| Button  | Clicking on this button opens a dialog box *(see page 412)* which you can use to refine the filter. |
| Name | Column in which you enter the name (symbol) of the instance or data type that you want to display. You can use wild cards (*** or**). |
| EDT | Authorizes the display of the Elementary Data Types. |
| DDT | Authorizes the display of Derived Data Types. |
| IODDT | Authorizes the display of the derived data concerning inputs and outputs. |
| Name | Column in which the name (symbol) of the instance or data type is located. |
| Type | Column in which the data type is located. |
| Comment | Column in which the data comment is located. |

### Function Blocks tab

The table below describes the elements of the **Function Blocks** tab of the data selection tool.

| Field | Description |
|---|---|
| Button  | Clicking on this button opens a dialog box *(see page 412)* which you can use to refine the filter. |
| Name | Column in which you enter the name (symbol) of the instance or data type that you want to display. You can use wild cards (*** or**). |
| Named by default | Authorizes the display of data named by default |
| EFB | Authorizes the display of Elementary Function Block data types. |
| DFB | Authorizes the display of Derived FB data types. |
| Name | Column in which the name (symbol) of the instance or data type is located. |
| Type | Column in which the data type is located. |
| Comment | Column in which the data comment is located. |

# Section 48.5
## Manipulation of objects in runtime screens

### Subject of this section

This section describes how to manipulate objects in the runtime screens. how to select objects, move objects, duplicate or delete objects,...

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| How to select graphic objects | 1646 |
| How to move and resize objects | 1648 |
| How to delete and copy objects | 1650 |
| How to position objects | 1652 |
| How to create compound objects | 1654 |

# How to select graphic objects

## At a Glance

When objects have been created in a screen, you may have to select some of them in order to position them, regroup them, etc. The selection or de-selection of graphic objects is one of the basic tasks during the construction of an operator screen.

## How to select a single object

The following table describes the operations to perform in order to select an object.

| If you want to select | Then |
|---|---|
| an isolated object. | validate selection mode then click on the object with the left mouse button. |
| an object among several overlaid objects. | validate the selection mode, press **ALT** then right mouse click several times on the area occupied by the objects.<br>**Result**: You select each object one after the other even they are hidden (in the background). |
| successive objects on a screen. | press the space bar successively.<br>**Result**: Each time you press the spacebar, you select the next object even if it is hidden behind another object. |

## How to select several objects

The following table describes the operations to perform in order to select several objects at the same time.

| If you want to select | Then |
|---|---|
| several contiguous objects. | frame the objects to be selected by defining a selection zone (this operate mode is identical to that for the creation of objects).<br>**Result**: A rectangle with handles is displayed on the screen. It regroups all the objects located in the selection zone. |
| several objects scattered in the operator screen. | press **Shift** then click on the objects to be selected while keeping the **Shift** key held down.<br>**Result**: A rectangle with handles is displayed on the screen, it groups all the selected objects. |

### How to deselect a selected object or a group of objects

The following table describes the operations to perform in order to deselect an object.

| Step | Action |
|---|---|
| 1 | Press **Shift** and hold down this key. |
| 2 | Left click on the selected object that you want to deselect.<br>**Result**: The selection handles associated with the object disappear, indicating that the object is no longer part of the selection. |

**NOTE:** Press **ESC** to delete the entire active selection. Clicking on the screen background gives the same results.

# How to move and resize objects

## Introduction

You may have to move or resize objects created in a screen, if any .

## How to use the mouse to move objects

The following table describes the operations to perform in order to move one or several objects with the mouse.

| Step | Action |
|---|---|
| 1 | Select the object or objects to be moved. |
| 2 | Click with the left mouse button on a selected objects or in the rectangle representing the selection. |
| 3 | Keep the button held down. |
| 4 | Drag the selected object or the rectangle to the desired position.<br>**Result**: the selection is represented by a dotted zone as you move it. |
| 5 | Release the mouse button.<br>**Result**: the selection is placed in its new location. |

## How to use the keyboard to move objects

The following table describes the operations to perform in order to move one or several objects with the keyboard.

| If you want to move | Then |
|---|---|
| precisely one or several objects by using the keyboard | after having selected one or several objects to move, press as many times as necessary on the keyboard direction (arrow) keys.<br>**Result**: the selection represented by a dotted zone can be moved pixel by pixel or according to the steps of the grid if it is active. |
| rapidly one or several objects by using the keyboard | after having selected one or several objects to move, press on the **Shift** key, then while holding down this key press as many times as necessary on the keyboard direction (arrow) keys.<br>**Result**: the selection represented by a dotted zone can be moved in 10 pixel-steps. |

## How to resize objects

The following table describes the operations to perform to resize one or several objects.

| Step | Action |
|---|---|
| 1 | Select the object or objects to be modified |
| 2 | Drag the handles in the desired directions in order to resize the frame.<br>**Result**: the selection represented by a dotted zone is resized as you move the mouse. |

# How to delete and copy objects

## At a Glance

You may need to delete a graphic object or duplicate the same object in a screen.

## How to delete objects

The following table describes the operations to perform in order to delete one or several objects.

| Step | Action |
|---|---|
| 1 | Select the object or objects to be deleted.<br>**Result**: The selection is represented by a frame with handles. |
| 2 | Press **Delete**.<br>**Result**: The selected objects are deleted. They are not memorized in the clipboard. |

## How to use the mouse to copy objects

The following table describes the operations to perform in order to copy one or several objects with the mouse.

| Step | Action |
|---|---|
| 1 | Select the object or objects to be copied. |
| 2 | Click with the left mouse button on one of the selected objects or in the rectangle representing the selection. |
| 3 | Hold down the button and press **CTRL**. |
| 4 | Drag the selected object or the rectangle to the desired position.<br>**Result**: The selection is represented by a dotted zone as you move it. |
| 5 | Release the mouse button.<br>**Result**: The copy of the selection is placed in its chosen location. |

## How to cut, copy or paste objects

The table below indicates the procedures for cutting, copying or pasting one or several objects.

| Step | Action |
|---|---|
| 1 | Select the object or objects to manipulate.<br>**Result**: The selection is represented by a frame with handles. |
| 2 | Select the command to execute:<br>● **Edit → Cut**<br>● **Edit → Copy**<br>● **Edit → Paste**<br>**Result**: **Cut** deletes the selected item(s) and memorizes it (them) on the clipboard. **Copy** duplicates the selected items in the clipboard and **Paste** duplicates the clipboard contents on the screen. |

# How to position objects

## At a Glance

You may have to position an object in relation to another in a screen:

- to align the objects,
- to make objects symmetrical,
- to position the objects in the foreground and the background in relation to the others.

## How to align a group of objects

The table below describes the procedure for aligning a group of objects.

| Step | Action |
|---|---|
| 1 | Select the objects to align.<br>**Result**: The selection is represented by a frame with handles. |
| 2 | Select the command to execute in the toolbar:<br>● Left alignment,<br>● Right alignment,<br>● Upper alignment,<br>● Lower alignment.<br>**Result**: The selected objects are aligned according to what has been chosen. |

## How to make an object or a group of objects symmetrical

The table below describes the procedure for flipping an object or a group of objects vertically or horizontally.

| Step | Action |
|---|---|
| 1 | Select the object or objects to be flipped.<br>**Result**: The selection is represented by a frame with handles. |
| 2 | Select the flip command in the toolbar:<br>● vertical flip,<br>● horizontal flip.<br>**Result**: The selected object or group of objects is positioned according to the chosen symmetry. |

## How to put an object in the foreground

The table below describes the procedure for placing an object in the foreground.

| Step | Action |
|---|---|
| 1 | Select the object to place in the foreground.<br>**Result**: The selected object is equipped with handles. |
| 2 | Select the **Foreground** command in the toolbar.<br>**Result**: The selected object is positioned in the foreground. |

## How to put an object in the background

The table below describes the procedure for placing an object in the background.

| Step | Action |
|---|---|
| 1 | Select the object to place in the background.<br>**Result**: The selected object is equipped with handles. |
| 2 | Select the **Background** command in the toolbar.<br>**Result**: The selected object is positioned in the background. |

# How to create compound objects

## At a Glance

The objects of a screen are sometimes associated to become a single compound object (that can be animated). Similarly, you may have to unbind a compound object into several simple objects, in order to manipulate them individually.

## How to Bind a group of objects

The table below describes the procedures for binding a group of objects.

| Step | Action |
|---|---|
| 1 | Select the objects to bind.<br>**Result**: The selection is represented by a frame with handles. |
| 2 | Activate the **Group** command.<br>**Result**: The objects are grouped in a single **compound object**. the handles belonging to the selected objects disappear and a rectangle containing all of the selected objects and represented by handles is displayed. |

## How to Unbind a group of objects

The table below describes the operations to perform to unbind a group of objects.

| Step | Action |
|---|---|
| 1 | Select the compound object to unbind.<br>**Result**: The selected object is represented by handles. |
| 2 | Activate the **Ungroup** command.<br>**Result**: All the objects contained in the compound object are viewed with their handles. If the object is animated, the animated characteristics are lost. |

## How to select an element belonging to a compound object

The table below describes the operations to perform to modify the characteristics of an item belonging to a compound object, without having to ungroup it.

| Step | Action |
|---|---|
| 1 | Select the compound object.<br>**Result**: The selected object is represented by handles. |
| 2 | Press **Alt** then right click on the compound object, while holding down the **Shift** key.<br>**Result**: The elements comprising the compound object are selected successively. The selected object can not be manipulated as a simple object. |

# Section 48.6
## Managing the runtime screens messages editor

### Subject of this section

This section describes how to manage the runtime screens messages editor.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Operator screens messages editor | 1656 |
| How to manage messages linked to operator screens | 1658 |
| Find dialog box | 1660 |
| Replace dialog box | 1662 |
| Message Renumbering Dialog Box | 1663 |
| Message editing and import/export function | 1664 |

# Operator screens messages editor

## At a Glance

Messages comprise a text and a label. They are memorized in the project and are linked to objects via their number. In this way, it is not necessary to create a message per object and duplicate identical messages.

When a project is created, the message list is empty, and the user needs to fill it in: by creating your own messages or by importing messages from another file or another tool.

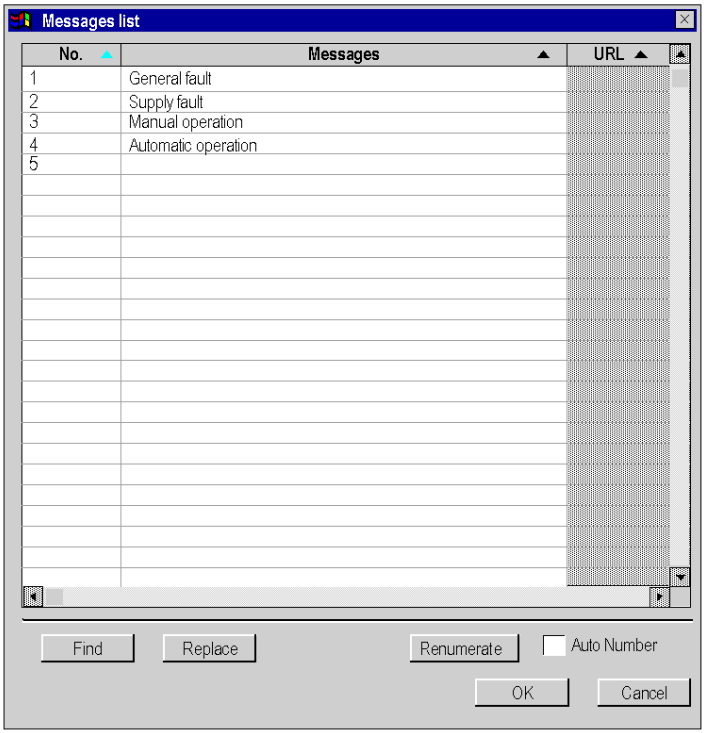
## Displaying the message list

The following table describes the operations to perform in order to display the message list.

| Step | Action |
|---|---|
| 1 | Select the **Operator screens** directory from the project browser. |
| 2 | Select the **Message list** → **Open** command in the contextual menu:<br>**Result**: The **Message list** window appears on the screen. |

## Message screen

The screen below presents an example of a message list:




## Description

The following table describes the commands accessible from the message screen.

| Command | Description |
|---|---|
| **Find** | Allows you to search for a string of characters (or a message number). |
| **Replace** | Allows you to replace a string of characters (or a message number). |
| **Auto Number** | Allows automatic dialing of messages. |
| **Renumerate** | Gives access to a dialog box which allows you to define the first automatic dial number. |

# How to manage messages linked to operator screens

## At a Glance

The dialog box for managing messages allows you to create, modify or delete messages in addition to linking them to a number or a hyperlink.

## How to create a message

The table below describes the operations performed to create a message to be used by the operator screen editor.

| Step | Action |
|---|---|
| 1 | Right click on the **Operator Screens** directory in the project browser.<br>**Result:** The relevant contextual menu appears. |
| 2 | Click **Messages list** → **Open**.<br>**Result:** The messages management dialog box appears. |
| 3 | Enter a message number in the **No.** column If you have checked the **Auto number** box, the number will be automatically entered, when the message is validated.<br>**Result:** The number entered appears in the list. |
| 4 | Enter your message in the **Messages** column, then press **Enter** to validate. The maximum length of a message is 255 characters.<br>**Result:** Your message and its associated number appear in the list. |
| 5 | Repeat steps 3 and 4, to enter the other messages on the list. Click **OK** to confirm the command.<br>**Result:** The messages management dialog box closes and the list is memorized. |

### How to link a hyperlink to a message

The table below describes the operations performed to link a hyperlink to a message.

| Step | Action |
|---|---|
| 1 | Open the message list, |
| 2 | Right click on the **URL** message column.<br>**Result:** The relevant contextual menu appears. |
| 3 | Activate the **Create** command.<br>**Result**: The **Insert hyperlink** window appears. |
| 4 | Enter the name with the document path or the WEB page to link to the message. Or select it with the **Browse** button. |
| 5 | Click **OK** to validate the hyperlink.<br>**Result:** The hyperlink appears in the **URL** column. |
| 6 | Click **OK** to confirm the command.<br>**Result:** The messages management dialog box closes and the list is memorized. |

### How to modify a message

The following table describes the operations performed to modify an existing message.

| Step | Action |
|---|---|
| 1 | Right click on the **Operator Screens** directory in the project browser.<br>**Result:** The relevant contextual menu appears. |
| 2 | Click **Messages list** → **Open**.<br>**Result:** The message screen appears. |
| 3 | Right click on the message to be modified.<br>**Result**: A cursor appears in the text. |
| 4 | Move the cursor to where you want to modify your text. To delete a word, double click on it to select it and press **Del**. Enter the new text or the new number then **Enter** to validate.<br>**Result:** The message is modified. |
| 5 | After you complete modifications, click **OK** to validate.<br>**Result**: The new messages are backed up in the project. |

**NOTE:** Press **ESC** to delete all the modifications made in the current session.

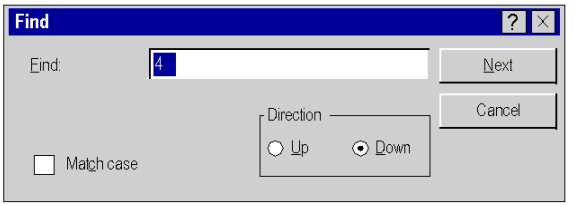
# Find dialog box

## At a Glance

The **Find** dialog box allows you to find a string of characters or a number in the list of messages.

## Dialog box

The screen below allows users to set the find parameters.




## Description

The following table describes the different fields offered by the dialog box.

| Field | Description |
|---|---|
| **Find** | Allows you to enter the string of characters (or number) you want to find. |
| **Match case** | If this box is checked, find will match the case of the entered characters (upper or lower case). |
| **Direction** | This field allows you to define the search direction in the list: move up or move down. |
| **Next** | This command initializes find or starts the next find. |
| **Cancel** | This command cancels the search. |

### How to find and replace a string of characters or a number.

Follow the steps below to find or replace a string of characters (or a number) in the list of messages:

| Step | Action |
|---|---|
| 1 | Right click on the **Operator Screens** directory in the project browser.<br>**Result**: The relevant contextual menu appears |
| 2 | Click **Messages list** → **Open**.<br>**Result**: The message manager dialog box appears. |
| 3 | Enable the **Find** command to find a string of characters or **Replace** to find and replace a string of characters.<br>**Result**: A **Find** dialog box appears |
| 4 | Fill in the different fields proposed by the dialog box and start the find or replace *(see page 1662)* of the string of characters.<br>**Result:** The changes are integrated into the list of messages. |
| 5 | After you complete modifications, click **OK** to validate.<br>**Result**: The new messages are backed up in the project. |

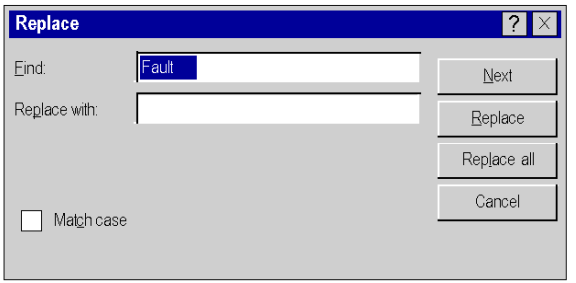
# Replace dialog box

## At a Glance

Dialog box **Replace** allows you to find a string of characters (or a number) in the list of messages and to replace it with another string of characters (or a number). You may select the item to replace or replace all.

## Dialog box

The screen below allows users define the replace parameters.




## Description

The following table describes the different fields offered by the dialog box.

| Field | Description |
|---|---|
| **Find** | Allows you to enter the string of characters (or number) you want to find. |
| **Replace** | This field allows you to enter the string of characters (or number) you want to replace. |
| **Match case** | If this box is checked, find will match the case of the entered characters (upper or lower case). |
| **Next** | This command initializes find or starts the next find. |
| **Replace** | This command replaces the string of characters found with the new string of characters that you defined in the **Replace with** field. |
| **Replace all** | This command replaces the string of characters that you defined throughout the list of messages. |
| **Cancel** | This command cancels the find/replace command. |

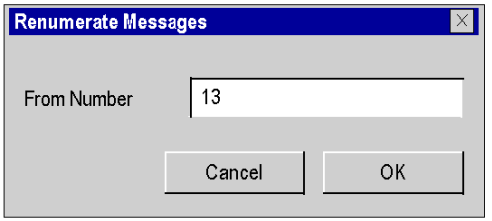
# Message Renumbering Dialog Box

## At a Glance

The renumbering dialog box allows you to change a number of one or several messages in the list automatically.

## How to Modify a Message Number

To modify a message number, you can proceed in the same way as when modifying a message or use the **Renumerate** command. The table below describes the procedure for modifying a message number using the **Renumerate** command.

| Step | Action |
|---|---|
| 1 | Right click on the **Operator Screens** folder in the project browser.<br>**Result**: The relevant contextual menu appears. |
| 2 | Execute the **Messages list** → **Open** command.<br>**Result**: The message manager dialog box appears. |
| 3 | Left click in the number field of the message to be modified.<br>**Result**: The field is selected. |
| 4 | Activate the **Renumerate** command.<br>**Result:** The following dialog box appears:<br>  |
| 5 | Enter the new number and confirm with **Enter**.<br>**Result**: The dialog box disappears and the selected message will be renumbered from the number entered in the **From Number** field. |
| 6 | After you complete modifications, click **OK** to validate.<br>**Result**: The new numbers are backed up in the project. |

# Message editing and import/export function

## At a Glance

The operator screen messages also have edit and import/export functions.

## Information on editing functions

The editing functions are accessed by the commands:

- **Edit → Cut**
- **Edit → Copy**
- **Edit → Paste**
- **Edit → Delete**

**NOTE:** The paste function also allows messages copied in tools such as Excel or Word to be pasted directly into the message screen.

They are also accessible via the contextual menu: right click on the message.

There are also other functions in the contextual menu from the messages list.

| Function | Description |
|---|---|
| Cancel | Cancels the last completed action. |
| Redo | Repeats the last completed action. |
| Clear message | Erases the content of the selected row. |
| Insert row | Inserts a blank row above the selection: |
| Delete row | Deletes the selected row. |

You can **sort** the messages list by:

- the message number (increasing or decreasing order),
- alphabetical order (increasing or decreasing) of messages,
- alphabetical order (increasing or decreasing) of hyperlinks.

To start sorting, just click on the heading of the column you want to sort.

## How to import a list of messages

The table below describes the procedure for importing a list of messages.

| Step | Action |
|---|---|
| 1 | Right click on the **Operator Screens** directory in the project browser.<br>**Result**: The relevant contextual menu appears. |
| 2 | Execute the **Messages list** → **Import** command.<br>**Result**: A dialog box is displayed asking you to choose the file to be imported. |
| 3 | Select the file to be imported (directory tree and file) then confirm with the **Import** command.<br>**Result:** Another dialog box is displayed asking you to choose the type of import to be made. |
| 4 | Check the **Replace**[1] box, or<br>Check the **Merge (Based on new messages)**[2] box, or<br>Check the **Merge (Based on new numbers)**[3] box. |
| 5 | Click **OK** to confirm the modifications to the list.<br>**Result**: The new messages are backed up in the project. |

**Note**[1]: All currently defined messages in the project are deleted and then the new messages are imported.

**Note**[2]: The currently defined messages in the project are kept. When one of the new imported messages has the same message text as a message already existing in the project, this message is not imported. All the other imported messages are added at the end of the current message list of the project.

**Note**[3]: The currently defined messages in the project are kept. When one of the new imported messages has the same message number as a message already existing in the project, this message is not imported. All the other imported messages are added at the end of the current message list of the project.

**NOTE:**
To import a list of messages always use the

- **Operator Screens** → **Messages list** → **Import** command

and **NOT** the

- **Operator Screens** → **Import** command.

Otherwise the dialog box for selecting the **Import type** will not be displayed (see step 3 and 4 in the table above).

### How to export a list of messages

The table below describes the procedure for exporting a list of messages.

| Step | Action |
|---|---|
| 1 | Right click on the **Operator Screens** directory in the project browser.<br>**Result**: The relevant contextual menu appears. |
| 2 | Execute the **Messages list** → **Export** command.<br>**Result**: A dialog box appears to allow you to specify the name of the export file. |
| 3 | Choose the directory (directory tree) and enter the name of the export file (the extension.XCR is automatically added). Click on the **Export** command to confirm.<br>**Result:** Messages are exported in a text file, each line containing the number and label of each message. This file can then be used with any text editor. |

# Chapter 49
## Runtime screens in online mode

### Object of this chapter

This chapter describes how the runtime screens editor works in online mode.

### What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Basic functions in online mode | 1668 |
| How to select an animated object | 1669 |
| How to use command objects to modify the variables associated with objects | 1670 |
| How to view the variables associated with graphic objects | 1671 |
| How to modify the variables associated with objects from an animation table | 1672 |
| How to search for cross-references | 1673 |
| How to manage the display of operator screens | 1674 |
| How to modify screen control | 1676 |
| How to obtain information on objects when in connected mode | 1677 |

# Basic functions in online mode

## Introduction

After you create your operator screens, you must change to online mode, in order to monitor and control the automation process.

To do this, enable the relevant icon or **PLC** → **Connect** command

## Editing online

A screen can only be edited in online mode if you have not given priority *(see page 563)* to the **PLC driver**.

## Display rules for a simple object

Animated objects which make up a screen are displayed when their display condition has been met.

## Display rules for a compound object

When a compound object is animated but its display conditions have not been met, none of its component items are displayed (even if it is made up of items whose display requirements have been met).

## Basic points about display of related objects

The objects on a page are displayed according to the positioning order assigned to them by the **Foreground** or **Background** commands.

**NOTE:** Command objects are always positioned above other graphic objects.
To avoid control errors or display problems, we advise you do not overlay command objects.

# How to select an animated object

## At a Glance

You may need to select an object in online mode. We advise you to follow the procedure described below for the process to remain unchanged during the selection.

## How to select an animated object

The table below describes the procedure for selecting an object in online mode

| If you want to select | then |
|---|---|
| an animated object using the function keys. | press **F4** or **F5**.<br>**Result**: The **F4** key selects animated objects one after the other in the order that they were created or repositioned by the user. The **F5** key selects them in the reverse order. |
| an animated object using the tool bar. | click on the **next animated object** or **previous animated object** icons.<br>**Result**: It is identical to pressing the **F4** and **F5** keys. |
| a control command object. | press and hold down the **Ctrl** key, then left click on the object.<br>**Result**: The object is selected. It no longer commands the control and its properties can be modified. |
| any object. | press the **Space** bar.<br>**Result**: The **Space** bar selects all the objects one after the other according to how they were created or repositioned. Simultaneously press **Caps Lock + Space** to select them in the reverse order. |

# How to use command objects to modify the variables associated with objects

## At a Glance

In online mode, you may use the command objects in an operator screen to modify the variables associated with the animated objects. By default, you cannot modify the command objects when a screen is being opened. You must use the **F7** key or the **Services → Enable Variable Modification** command to allow these modifications.

## How to use a command object to modify variables

The following table describes the procedures for using a command object to modify variables.

| Step | Action |
|---|---|
| 1 | Open the operator screen in which the command objects are located. |
| 2 | Press the **F7** key or select the **Services → Enable Variable Modification** command. |
| 3 | Give control to the command object that enables the modification of the variable settings. |
| 4 | Modify the command object settings. |

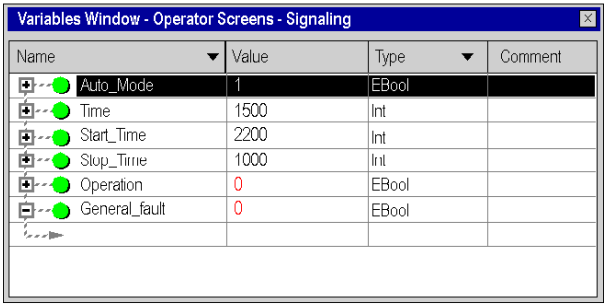
# How to view the variables associated with graphic objects

## At a Glance

In online mode you may view the settings of variables associated with an operator screen in the **Variables Window** tool. The **Variables Window** tool only allows you to view the variables. You cannot edit them.

## How to view the variables associated with graphic objects

The following table describes the operations necessary to display the variables associated with the graphic objects.

| Step | Action |
|---|---|
| 1 | Select the operator screen. |
| 2 | Select the **Tools** → **Variables Window** where the variables you want to view are located.<br>**Result**: The **Variables Window** appears<br>  |
| 3 | Double click on a variable in the **Variables Window**.<br>**Result**: The object associated with this variable appears in the operator screen, surrounded by a dotted frame. |

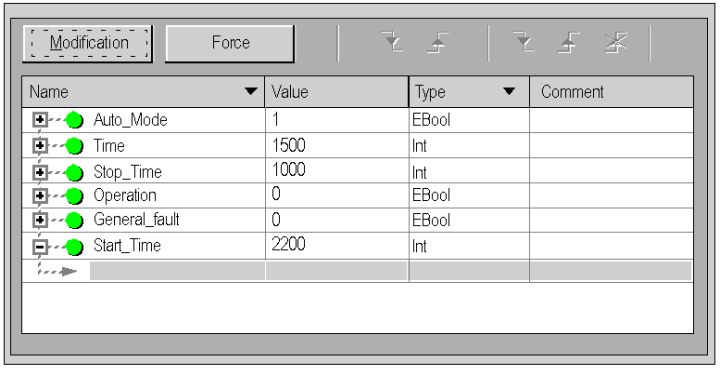
# How to modify the variables associated with objects from an animation table

## At a Glance

In online mode you may modify the variables associated with the animated objects from an animation table.

## How to modify the variables associated with objects from an animation table

The table below describes how to modify the variables associated with graphic objects from an animation table.

| Step | Action |
|---|---|
| 1 | Select the objects for which you want to modify the variables.<br>**Result**: The selected objects appear with handles and have a dotted frame. |
| 2 | Use the **Ctrl** + **T** shortcut key or enable the **Services** → **Initialize Animation Table** command.<br>**Result**: All the variables relating to the selected objects appear in a new animation table. You can now modify their value.<br>The diagram below presents an example of an animation table produced by selecting animated objects.<br>  |

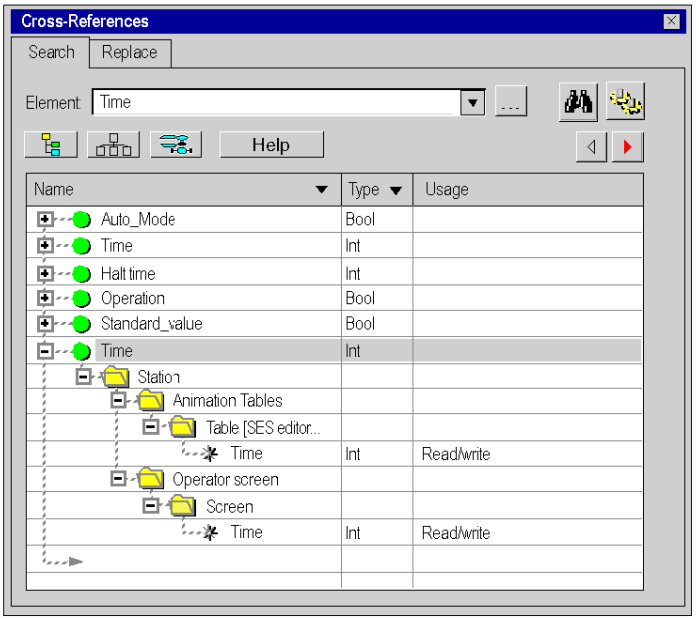
# How to search for cross-references

## At a Glance

In connected mode as in local mode, you can search in the project programs for the variables linked to an animated object. This is known as the cross references search.

## How to search for cross references

The table below describes the procedures for searching for cross references in an animated object.

| Step | Action |
|---|---|
| 1 | Select the object(s) in the operator screen.<br>**Result**: The selected objects appear with handles and have a dotted frame. |
| 2 | Use the **Ctrl** + **U** shortcut or click **Services** → **Initialize Search**.<br>**Result**: All the variables relating to the selected objects appear in the **cross references** search window.<br>The diagram below is an example of a cross reference search performed by selecting the animated objects.<br>  |

# How to manage the display of operator screens

## At a Glance

You may need to modify the display of screens in connected mode. You may for example:

- Mask the horizontal and vertical scrollbars,
- Go to **full screen**mode.

## How to mask the scroll bars

The table below describes the procedure for masking scrollbars displayed by the operator screens editor.

| Step | Action |
|---|---|
| 1 | Click **Tools** → **Options**. |
| 2 | Select the **operator screen** tab.<br>**Result**: The editor configuration *(see page 563)* dialog box is displayed. |
| 3 | Uncheck the **Horizontal scroll bar** and **Vertical scroll bar** and validate with **OK**.<br>**Result**: The graphic editor no longer has vertical and horizontal scrollbars. |

## How to switch to full screen mode

The table below describes the procedure for switching to **full screen** mode.

| Step | Action |
|---|---|
| 1 | Click **Display** → **Full screen**.<br>**Result**: The menu and toolbars disappear. |

## How to exit a full screen mode.

The table below describes the procedure for switching to full screen mode.

| Step | Action |
|---|---|
| 1 | Press **Esc** to exit the full screen mode.<br>**Result**: The screen changes from **full screen** mode to **window** mode. |

### How to start the operator screens editor automatically while in full screen mode

The table below describes the procedure for automatically starting the operator screens editor in full screen.

| Step | Action |
|---|---|
| 1 | Click **Tools** → **Project settings**. |
| 2 | Select the **operator screen** tab.<br>**Result**: The editor configuration *(see page 550)* dialog box is displayed. |
| 3 | Check the **Full screen at opening** box and validate with **OK**.<br>**Result**: The operator screens editor will be automatically started in full screen mode. |

# How to modify screen control

## At a Glance

The control type is determined by the configuration *(see page 550)* options. It can be modified in online mode.

**NOTE:** When the runtime screens are controlled via PLC it is not possible to edit them in online mode, even if the option **Enable Edit in online mode** is set via **Tools** → **Options** → **Operator Screens** tab.

## How to set the control type in online mode

The following table describes the procedure for changing the control type for operator screens.

| If you want | and if you are | then |
|---|---|---|
| the screens to be controlled by the PLC, | in operator control mode, | press **F2** or click on the **PLC controlled** icon. |
| the screens to be controlled by the operator, | in PLC control mode, | press **F3** or click on the **Operator controlled** icon. |

# How to obtain information on objects when in connected mode

## Introduction

In connected mode, you can obtain two types of information via the mouse:

- Is the object a driving object,
- which variable is associated to this object.

## How to ascertain whether an object is a driving object

The table below describes the procedure for ascertaining whether an object is a driving object.

| Step | Action |
|---|---|
| 1 | Position the mouse cursor on the object.<br>**Result:** if the object is a driving object, the cursor takes the shape of a hand. |

## How to find out the name and the value of a variable associated with an object

The table describes the procedure for finding out the name and value of a variable associated with an object.

| Step | Action |
|---|---|
| 1 | Position the mouse cursor on the object.<br>**Result:** the name of the variable and its current value in the PLC are displayed. |

# Chapter 50
## Runtime screens objects library

# Section 50.1
## Managing library objects

### Subject of this section

This section describes how to manage the library graphic objects.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| The Objects Library | 1681 |
| How to manage the list of graphic items in the library | 1682 |
| Properties of the object library and its elements | 1685 |
| Handling the Objects | 1689 |
| Dereferencing in Operator Screens | 1691 |

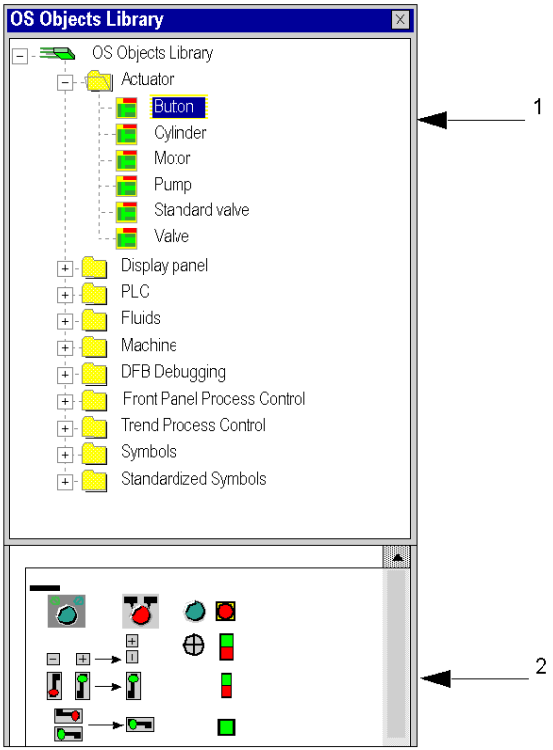
# The Objects Library

## At a Glance

The objects library provides manufacturer objects which can be inserted into the operator screens. The objects are arranged in families. The library allows operators to create their own objects and insert them into a library family.

The library may be opened with the **Tools** → **Operator Screen Library** command.

The figure below presents the objects library.




The table below provides a description of the objects library.

| Number | Description |
|---|---|
| 1 | The browser displays all the families and objects. |
| 2 | The window displays the graphic representation(s) of the selected object. |

# How to manage the list of graphic items in the library

## At a Glance

The library graphical objects are organized in families for optimum management. You cannot create new objects outside a family.

The main actions that can be performed on objects or object families are described below.

## How to create an object family in the library

The following table describes the steps to be carried out to create a new object family in the library.

| Step | Action |
|---|---|
| 1 | Execute the **Tools** → **Operator screens library** from the main Unity Pro menu<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Right click on the library root.<br>**Result:** The relevant contextual menu appears. |
| 3 | Enable the command **New family**.<br>**Result:** A new screen family appears in the objects library with a generic name. |

## How to create an object in the library

The following table describes the operations to be performed in order to create an object in the library.

| Step | Action |
|---|---|
| 1 | Click **Tools** → **Operator screens library** in the toolbar.<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Click with the right mouse button on an object family or library object.<br>**Result:** The relevant contextual menu appears. |
| 3 | Enable the **New object** command.<br>**Result:** A new object appears in the objects library with a generic name. |

### How to rename a library object or object family

The following table describes the operations to be performed in order to rename a library object family.

| Step | Action |
|---|---|
| 1 | Click **Tools** → **Operator screens library** in the toolbar.<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Click with the left mouse button on an object family or library object. |
| 3 | Press **F2**.<br>**Result:** The label is highlighted and the text input cursor flashes at the end of the input field. |
| 4 | Enter the new name of the object or family using the keyboard.<br>**Result:** The object or object family appears with its new name. |

### How to move an object in the library

The following table describes the operations to be performed in order to move an object from a family to another family in the library.

| Step | Action |
|---|---|
| 1 | Click **Tools** → **Operator screens library** in the toolbar.<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Left click on the library object to be moved.<br>**Result:** The relevant contextual menu appears. |
| 3 | Hold down the left mouse button to maintain the selection. |
| 4 | Move the object to the desired family while holding down the button.<br>**Result:** The object moves at the same time as the mouse cursor. |
| 5 | Release the mouse button.<br>**Result:** The object appears in the desired family. |

**NOTE:** You may move an object in the same family if the Manual sort *(see page 1685)* option is checked.

## How to open an object in the library

The following table describes the operations to be performed in order to open an object in the library.

| Step | Action |
|---|---|
| 1 | Click **Tools** → **Operator screens library** in the toolbar.<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Right click on a library object.<br>**Result:** The relevant contextual menu appears. |
| 3 | Activate the **Open** command.<br>**Result:** An operator screen window opens. It displays the graphic object or objects contained in the library object. |

## Information on editing functions

The editing functions are accessed by the commands:

- **Edit → Cut**
- **Edit → Copy**
- **Edit → Paste**
- **Edit → Delete**

Commands comply with Windows 32 bit ergonomics and apply to objects or families of objects in the library.

There is also a function for deleting the last completed action.

# Properties of the object library and its elements

## At a Glance

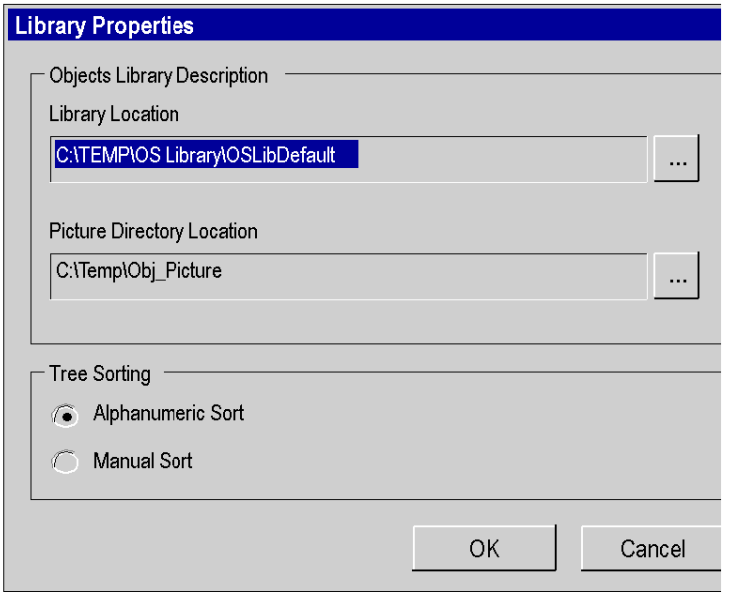
The objects library and its elements (family and objects) have properties. They allow the operator to modify:

- the access pathname of the library file,
- the access pathname of the library images directory,
- the type of family and object sorting in the library browser,
- the type of family and object sorting in the library browser,
- the name of families,
- the name of objects,
- the family linked to the object.

## Library properties

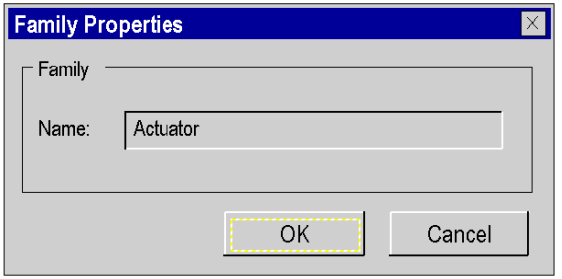
The table below presents the procedures for accessing the **Library Properties** dialog box.

| Step | Action |
|---|---|
| 1 | Execute the **Tools** → **Operator screens library** from the main Unity Pro menu.<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Select the library root in the browser application. |

| Step | Action |
|---|---|
| 3 | Select the **Properties** command from the contextual menu.<br>**Result:** The **Library Properties** window appears:<br>  |
| 4 | The access pathname and the filename (.bib extension) of the library are indicated in the **Library Location** field.<br><br>The  button allows the operator to browse the hard disk to change the pathname and the file. |
| 5 | The **Picture Directory Location** field contains the access pathname and file name (extension .bib) of the directory where the images used by the library are located.<br><br>The  button allows the operator to browse the hard disk to change the pathname and the storage directory name. |
| 6 | In the **Tree Sorting** zone you may choose to sort the families and objects. Sorting can be:<br>● in alphabetical order, by checking the **Alphanumeric Sort** box,<br>● manual (user option), by checking the **Manual Sort** box. |

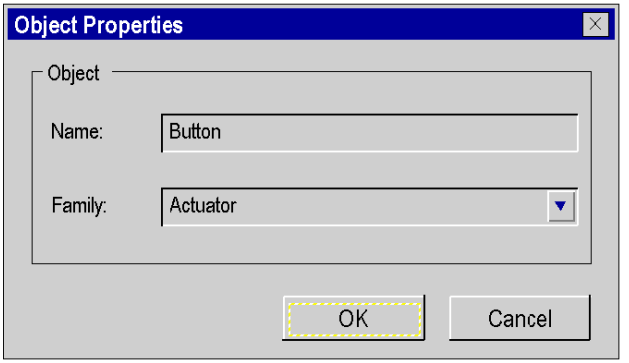
## Library properties

The table below presents the procedures for accessing the **Family Properties** dialog box.

| Step | Action |
|---|---|
| 1 | Execute the **Tools** → **Object Library** command from the Unity Pro main menu.<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Select a family in the library browser application. |
| 3 | Select the **Properties** command from the contextual menu.<br>**Result:** The **Family Properties** window appears:<br>  |
| 4 | In the **Name** field, give the family a name. It can be modified and can contain up to 255 characters. |

## Object properties

The table below presents the procedures for accessing the **Object Properties** dialog box.

| Step | Action |
|---|---|
| 1 | Execute the **Tools** → **Object Library** command from the Unity Pro main menu.<br>**Result:** The object library opens. The browser displays the objects and the families contained in the library. |
| 2 | Select an object in the library browser application. |
| 3 | Select the **Properties** command from the contextual menu.<br>**Result:** The **Object Properties** window appears.<br>  |
| 4 | In the **Name** field, give the object a name. It can be modified and can contain up to 255 characters. |
| 5 | The **Family** field displays the family with which the object is associated. Select the arrow on the right side of the field to associate the object with another family. |

# Handling the Objects

## At a Glance

When you open a library object, Unity Pro opens an operator screen window in edit mode. The operator screen contains the different graphic objects. From the operator screen, you can

- use all editing functions in the screen
- select and copy a graphic object in another screen
- bind variables to graphic objects

## Edit function

An object's operator screen editor works in the same way as the project operator screen editor. It has the same functions: The menu and toolbars *(see page 1580)* are identical.

## Copying an object

You can copy graphic objects between the operator screen of an object and:

- another object operator screen,
- a project operator screen.

In both cases the procedures are identical.

## Procedure

The table below describes the procedure for copying a graphic object in another library object or in a project operator screen.

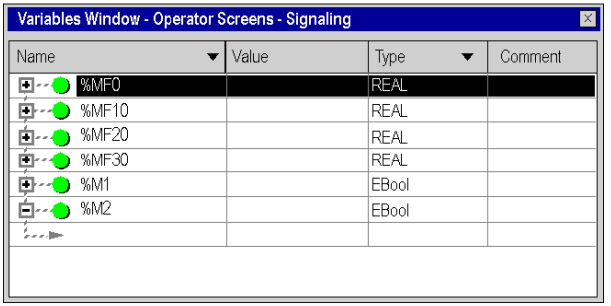
| Step | Action |
|---|---|
| 1 | Open *(see page 1684)* a library object.<br>**Result:** The operator screen with the graphic objects appears. |
| 2 | Right click on a graphic object.<br>**Result:** The relevant contextual menu appears. |
| 3 | Enable the **Copy** command. |
| 4 | Open a library object or an operator screen in which you want to copy the graphic object. |
| 5 | Right mouse click on the background of the operator screen. |
| 6 | Activate the **Paste** command.<br>**Result:** The graphic object appears in the operator screen. |

## Association of variable

Graphic objects are animated with variables that have topological addresses. Simply replace these variables with your project variables to animate the objects.

## Procedure

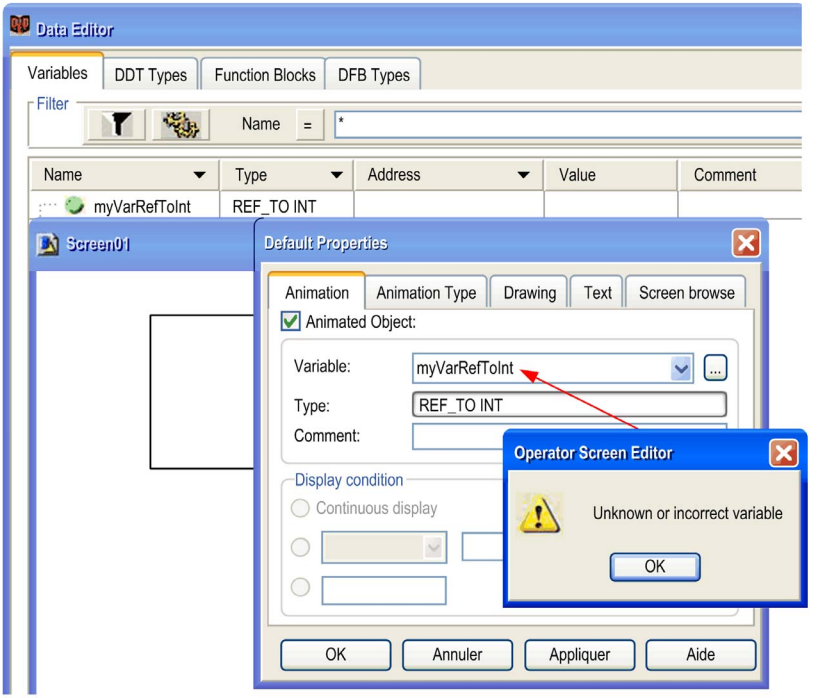
The following table describes the procedure for associating project variables with animation variables for graphic objects.

| Step | Action |
|---|---|
| 1 | Copy a graphic object from the library into an operator screen, then select it. |
| 2 | Select the **Tools** → **Variables Window** command.<br>**Result**: The **Variables Window** appears with all the variables of the graphic object<br>  |
| 3 | Double click on a variable in the **Variables Window**.<br>**Result**: The object associated with this variable appears in the operator screen, surrounded by a dotted frame. |
| 4 | Select the framed object. |
| 5 | Select the **Properties** command from the contextual menu.<br>**Result**: The **Object Properties** window appears. |
| 6 | Select the **Animation** tab.<br>**Result**: The variable that you selected initially appears in the **Variables** field. |
| 7 | Click on  to open the variables selection tool. |
| 8 | Select a project variable and click **OK**.<br>**Result**: The selected variable appears in the **Variables** field in the **Animation** tab and in the variables window. |
| 9 | For each of the project variables to be grouped, double-click on the variable in the variables window and repeat steps 7 and 8. |

# Dereferencing in Operator Screens

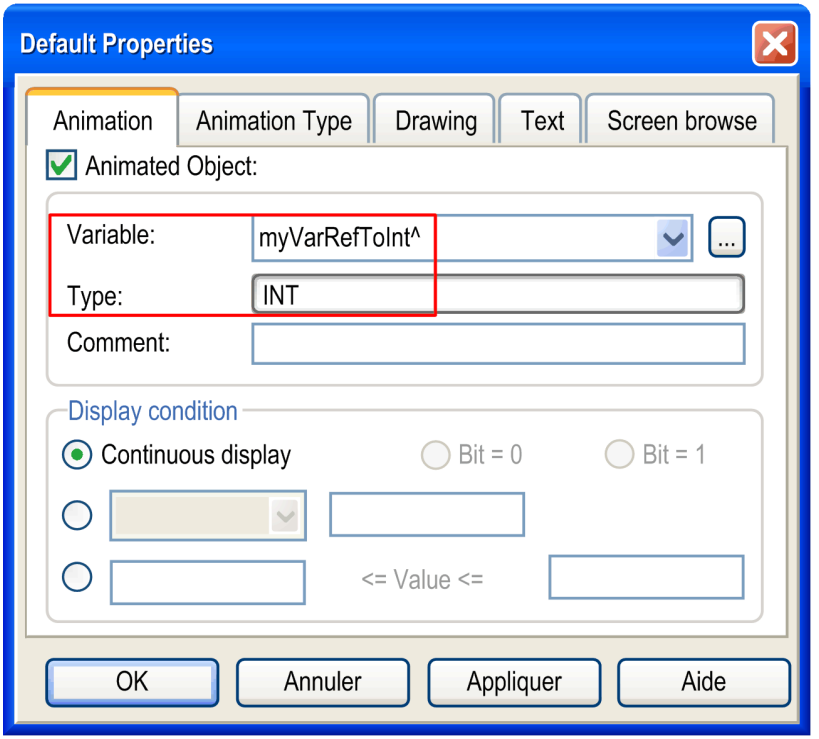
## References

References, with a REF_TO ANY type, cannot be entered in an operator screen:



## Dereferences

A dereferenced pointer (with the dereferencing operator “^”) can be entered; this is the actual variable itself:




**NOTE:** The dereferenced variable must have an Elementary Data Type.

**NOTE:** Only 1 level of dereferencing is allowed. This is not allowed: Myref^.MyInt^

# Part IX
## Documentation, Import/export

### Object of this part

This part presents the documentation and import/export functions.

### What Is in This Part?

This part contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| 51 | Documentation | 1695 |
| 52 | Import / Export | 1719 |

# Chapter 51
# Documentation

## Overview

This chapter describes how project documentation is created.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Creating Project Documentation | 1696 |
| Documentation Structure | 1697 |
| Topics in Documentation | 1700 |
| Creating Project Documentation | 1707 |
| Print Preview | 1709 |
| Page Setup | 1711 |
| Print | 1713 |
| Printout Setup | 1715 |

# Creating Project Documentation

## Introduction

Unity Pro lets you create the documentation for your project.

The subjects for the documentation are displayed in a tree structure. The structure contains all the project browser elements that were used. Elements that you have not used in the project do not appear as subjects in the documentation. The structure also contains some subjects specific to printing (for example the page title and table of contents).

In the tree structure, you can choose which subjects you want to print or display (print preview) and which you do not.

For some subject, parameters are available that you can use to define the printout.

For the printout you can choose between the **Structural View** or the **Functional View**.

You can print this documentation partially or fully after it has been created.

## Displaying the documentation structure

To display the documentation structure in the project browser you can either double-click **Documentation** directory, or right-click the **Documentation** directory and click **Open**.

# Documentation Structure

## General Information

The documentation structure *(see page 1707)* can be displayed with one of the two following views:

- Structural View *(see page 1698)*
- Functional view *(see page 1699)*

## Selecting the Type of Documentation Display

The following options are available to display the documentation structure in the structure view:

- Select the symbol in the tools list in the documentation structure.



- Select the subject in the documentation structure and execute the menu command **View** → **Structural View**.

The following options are available to display the documentation structure in the functional view:

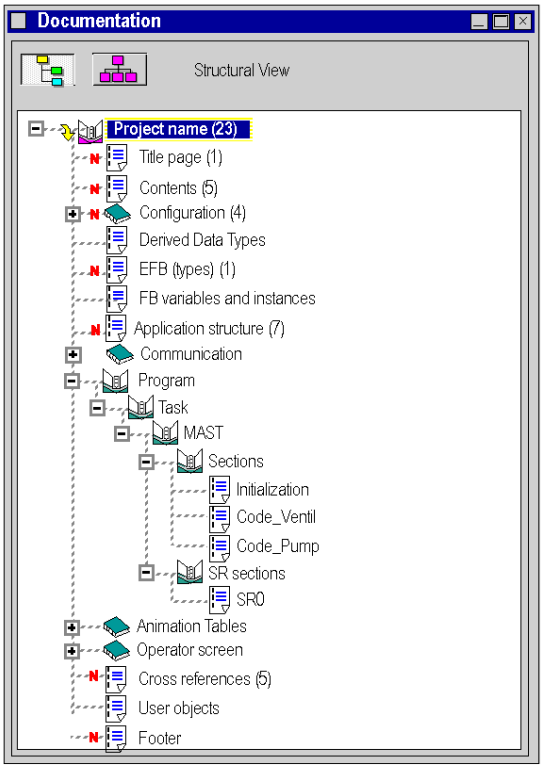
- Select the symbol in the tools list in the documentation structure.



- Select the subject in the documentation structure and execute the menu command **View** → **Functional View**.

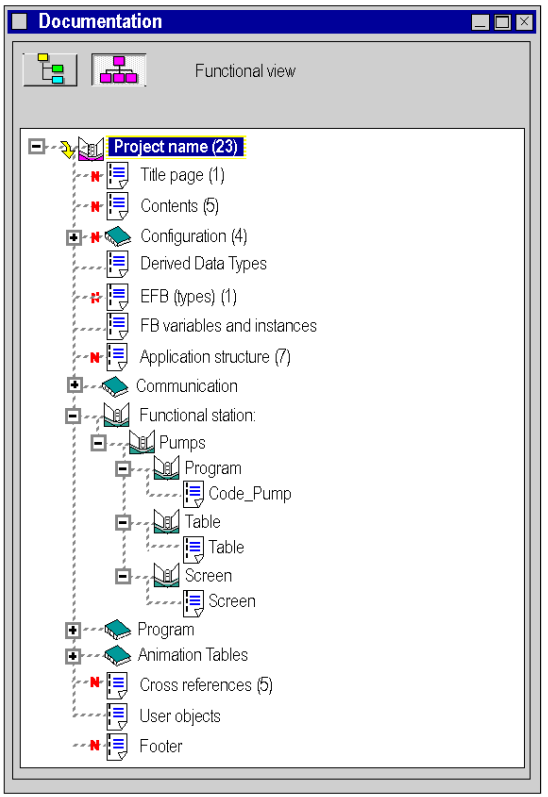
## Structural View

The documentation structural view displays the project structure in the form of objects. All objects are connected to their headings. That means that the sections and events are in the header program and the animation tables and user windows are in the header animation table or operator windows.



## Functional view

The documentation functional view displays the project structure in the form of functional units. Each functional unit contains the program, the animation table and the operator screens assigned to them. The objects that belong to a unit remain assigned to the appropriate headers in the structural view.



# Topics in Documentation

## Documentation Topics

The documentation folder of a project can contain the following topics:

- Title page
- Contents
- General information
- Configuration
- Derived Data Types
- Derived FB Types
- EFB types
- EF types
- FB variables and instances
- Application structure
- Functional station:
- Communication
- Program
- Animation table
- Operator screen
- Cross-References
- Footer

**NOTE:** Using the menu commands **Edit → Include heading** and **Edit → Exclude heading** you can define the topics you want to use in your documentation *(see page 1707)*.

## Title Page

This topic deals with printing the title page.

Topic contents:

- Name of the project
- Name of the designer
- Software version
- Creation date
- Date of last change
- Target PLC

**NOTE:** In the **Title Page** area of **Printout Setup** dialog box you can define the name of the project and the designer. All other details are automatically generated.

### Contents

The contents page is generated automatically according to the topics selected.

When a topic is selected, its corresponding number of pages appears in the contents. If the topic contains documentation but is not selected, **No print job** is displayed instead of the number of pages.

The total number of pages is displayed at the end of the contents.

### General Information

This topic deals with printing user content entered in **Documentation** → **General Information** tab of the project browser.

This information can be comments on the project, hyper-links and/or bitmaps.

### Configuration

This topic deals with printing the hardware configuration.

Topic contents:

- Bus topology
- Bus parameters (except for the X bus)
- PLC and rack configuration
- Parameterization of the input/output modules and the communication modules.

### Derived Data Types

This topic deals with printing the derived datatypes for the project.

Topic contents:

- Names of the data structure and their elements
- Data structure types (for example, structure, array) and element datatypes
- Data structure comments

### Derived FB Types

This topic deals with printing the DFBs used in the project.

Every DFB has a structure which contains the section of the DFB.

Topic contents:

- Topic **Derived FB types**
  - Table of all DFBs used in the project and their properties (name, version, creation / change date)
- Topic **"DFB name"**
  - Type name
  - Version
  - Short description of how the DFB work (if available)
  - Properties of the inputs, outputs, in/outputs and public variables (name, data type, initial value and comment)
- Topic **Sections**
  - Name of the section and the DFB
  - Section comment
  - Type of protection
  - Assigned functional module
  - Name of the condition for conditional call
  - Section code (except for the diagnostics DFBs provided by Schneider).
    Details on the section documentation can be found in the topic **Program**.

### EFB (Types)

This topic deals with printing the Elementary Function Block types used in the project.

Topic contents:

- Table of all EFBs used in the project and their properties (name, version, creation / change date)
- For every EFB type
  - Type name
  - Version
  - Short description of the how the EFB works
  - Properties of the inputs, outputs, in/outputs and public variables (name, data type, initial value and comment)

### FB (types)

This topic deals with printing the Elementary Function types (FBs) used in the project.

Topic contents:

- Table of all FBs used in the project and their properties (name, version, creation / change date)
- For every FB type
  - Type name
  - Properties of the inputs, outputs, in/outputs and public variables (name, data type, initial value and comment)

### Variables & FB instances

This topic deals with printing the following objects in a tabular format:

- Elementary datatypes (EDTs)
- Derived Data Types (DDTs)
- I/O derived data types (IODDTs)
- Instances of elementary functions
- Instances of elementary functions blocks
- Instances of derived functions blocks

Topic contents:

- Symbolic name (variable name or FB instance name)
- Constants (only for variables)
- Address (only for variables)
- Initial value
- Number of connections
- Global file (only for elementary datatypes and FB instances)
- Which attributes, for example, alias of, diag and so on (only if defined)

**NOTE:** In the **Variables & FB-Instances** area of **Printout Setup** dialog box you can define the attributes printed for the variables and FB instances.

### Application Structure

This topic deals with printing the project structure.

Topic contents:

- List of the program sections:
  - Release conditions
  - Comments
  - Module
  - Language type
- Sub-programs and macro steps calling tree

Depending on the view selected the documentation is shown in the **Structural View** or the **Functional View**. If the **Functional View** is chosen the section documentation is sorted according to the assigned functional module. The documentation of sections not assigned to any functional module is shown in its own table.

### Functional Station

This topic is only available in the **Functional View** and is used to print the sections assigned to a functional module.

Sections not assigned to documentation are not documented under the topic **Program**.

### Communication

This topic deals with printing the networks communication parameters.

## Program

This topic deals with printing the sections.

Topic contents:

- MAST task
  - Properties (for example, configuration, configuration task duration, watchdog time) in the MAST task.
  - Sections of the MAST task
  - Subroutine section of the MAST task
- FAST task
  - Properties (for example, configuration, configuration task duration, watchdog time) in the FAST task.
  - Sections of the FAST task
  - Subroutine section of the FAST task
- Events
  - Sections of the **Timer Events** tasks
  - Sections of the I/O event tasks

The contents of the section documentation depend on their programming languages.

**All sections:**

- Name of the sections and the assigned tasks
- Section comment
- Type of protection
- Assigned functional module
- Name of the condition for conditional call
- Section code

**FBD sections:**

- Component tables with **Execution by** attribute.
  Blocks that have an **Execute by** reference to another block appear in a table that shows the instance names and the position of the referenced and referencing blocks.

**LD sections:**

- Table for all of the object names not fully shown.
  Because of the fixed column width it can happen that variable names or block types are not shown fully. The incomplete object names appear in a table that shows the full object names and their position(s).
- Using the checkbox **Mixed Display Mode** in the **Printout Setup** dialog, you can define whether the documentation in the LD section is displayed in Input mode *(see page 720)* or in mixed display mode *(see page 721)*.

**SFC sections:**

- Tables for the step / action properties:
  - Step name
  - Position of the step object
  - Minimum and maximum supervision time
  - Delay Time
  - Step comment
  - Identifiers for the action(s)
  - Time literal / variable for the identifier (if required)
  - Action name
- Transition properties table:
  - Name of transition
  - Type of transition condition
  - Position of the transition object
  - Comment
- Jump properties table:
  - Target step name
  - Position of the jump object
  - Comment

### Animation Tables

This topic deals with printing the animation tables.

Topic contents:

- Name of the animation table
- Comment
- Functional module
- Name of the object (for example, variable name, EFB/DFB/DDT instance name, address)
- Type
- Comment

### Operator Screen

This topic deals with printing the operator screens.

This heading is used to print the properties of the operator screens, their graphic representations and the lists of messages associated with each screen.

Topic contents:

- Message lists
- Operator screen
  - Properties
  - Graphical representation
  - List of the animated variables

**NOTE:** In the **Operator Screens** area of **Printout Setup** dialog box you can define the scope and display of the documentation and the variables and FB instances.

### Cross-References

This topic deals with printing the cross-references for the application and the DFBs used.

Every cross-reference can contain the following tables:
- Addresses
- FB variables and instances
- FB objects
- Subroutine

Every table contains the following attributes:
- Name of the object
- Location of the reference (for example, section name, module name)
- Position used (for example, line and column in the section, type of module)

### User Objects

This topic deals with printing the properties of the hyperlinks and the user directories.

Contents of the topic **Hyperlinks**:
- Nodes containing hyperlinks (for example, station\animation tables)
- Comment
- Target path

Contents of the topic **User directories**:
- Nodes containing hyperlinks (for example, station\animation tables)
- Comment

### Footer

This topic deals with printing the footer.

Topic contents:
- Bitmap (logo)
- Author
- Dept.
- Target PLC
- Page title
- Two spare fields
- The date of printing
- Page number

**NOTE:** The target PLC, the page title, the printing date and the page number are automatically generated. In the **Footer** area of the **Printout Setup** dialog box you can define other attributes.

# Creating Project Documentation

## Creating Project Documentation

Carry out the following steps to create project documentation:

| Step | Action |
|---|---|
| 1 | Open the documentation structure. *(see page 1707)* |
| 2 | Select the documentation view *(see page 1707)*. |
| 3 | Select the individual subjects that you want to include or exclude (**Include heading** or **Exclude heading**). |
| 4 | Set up the page for printing *(see page 1711)*. |
| 5 | Define the parameters for the printout *(see page 1715)*. |
| 6 | Prepare the subjects for printing *(see page 1708)*. |
| 7 | Check the settings with the print preview *(see page 1709)*. |
| 8 | Select a printer and start printing *(see page 1713)*. |

## Opening the Documentation Structure

To open the documentation structure choose one of the two following options:

- Double-click **Documentation** directory in the project browser.
- Right-click the **Documentation** directory and click **Open**.

## Selecting the Documentation View

After opening the documentation structure you can choose between the Structural View *(see page 1698)* and the Functional View *(see page 1699)*.

## Including a Subject in the Documentation

You have the following options to include a subject in the documentation:

- In the documentation structure, right-click the subject to include and click **Include heading**.
- In the documentation structure, select the subject to include and execute the menu command **Edit → Include heading**.
- In the documentation structure, click beside the subject symbol to toggle between **Include heading** and **Exclude heading**.

Subjects included in the documentation are marked with a red symbol in front of the subject.

### Excluding a Subject From the Documentation

You have the following options to exclude a subject from the documentation:
- In the documentation structure, right-click the subject to exclude and click **Exclude heading**.
- In the documentation structure, select the subject to exclude and execute the menu command **Edit → Exclude heading**.
- In the documentation structure, click beside the subject symbol to toggle between **Include heading** and **Exclude heading**.

### Adding all Subjects in the Documentation

You have the following options to include all the subjects in the documentation:
- In the documentation structure, right-click the project and click **Include all headings**.
- In the documentation structure, select the project and execute the menu command **Edit → Include all headings**.
- In the documentation structure, click beside the project symbol to toggle between **Include all headings** and **Exclude all headings**.

Subjects included in the documentation are marked with a red symbol in front of the subject.

### Excluding all Subjects From the Documentation

You have the following options open to exclude all subjects from the documentation:
- In the documentation structure, right-click the project and click **Exclude all headings**.
- In the documentation structure, select the project and execute the menu command **Edit → Exclude all headings**.
- In the documentation structure, click beside the project symbol to toggle between **Include all headings** and **Exclude all headings**.

### Updating and Preparing

The documentation is updated dynamically whenever a change is made in the project directory.

After every update the updated or new subjects are marked with a **(?)** symbol.

You have the following options to prepare the project for printing:
- In the documentation structure, right-click the project and click **Generate Printout**.
- In the documentation structure, select the project and execute the menu command **Edit → Generate Printout**.

The subjects correctly prepared have the page number **(n)** displayed after the subject.

**NOTE:** It is only when the project subject has a page number that all the subjects are prepared.

**NOTE:** Printable documents are generated for every subject during preparation for printing. By default the documentation is not saved, it is deleted at the end of the Unity Pro session and must be generated again in the next Unity Pro session. If you want to use these documents in other sessions, check the **Save Generated Printout** checkbox in the **Printout Setup** dialog box.

# Print Preview

## Introduction

You can call up a print preview for every subject.

You have the following options to open the print preview:

- In the documentation structure, right-click the desired subject and click **View**.
- In the documentation structure, select the desired subject and execute the menu command **View** → **View**.

## Previous Page

You have the following options to show the previous page:

- In the documentation structure, right-click the displayed page and click **Previous Page**.
- In the documentation structure, select the desired displayed page and execute the menu command **Edit** → **Previous Page**.

## Next Page

You have the following options to show the next page:

- In the documentation structure, right-click the displayed page and click **Next Page**.
- In the documentation structure, select the desired displayed page and execute the menu command **Edit** → **Next Page**.

## Expand View

You have the following options to expand the view:

- In the documentation structure, right-click the displayed page and click **Zoom In**.
- In the documentation structure, select the desired displayed page and execute the menu command **Edit** → **Zoom In**.

## Contract View

You have the following options to contract the view:

- In the documentation structure, right-click the displayed page and click **Zoom Out**.
- In the documentation structure, select the desired displayed page and execute the menu command **Edit** → **Zoom Out**.

## View Full Page

You have the following options to show the whole page:

- In the documentation structure, right-click the displayed page and click **Zoom to Fit**.
- In the documentation structure, select the desired displayed page and execute the menu command **Edit** → **Zoom to Fit**.

### Close

You have the following options to close the print preview:

- In the documentation structure, right-click the displayed page and click **Close**.
- In the documentation structure, select the desired displayed page and execute the menu command **Edit** → **Close**.
- Press the **Esc** key.

### Print

Refer to the print section *(see page 1713)*.

# Page Setup

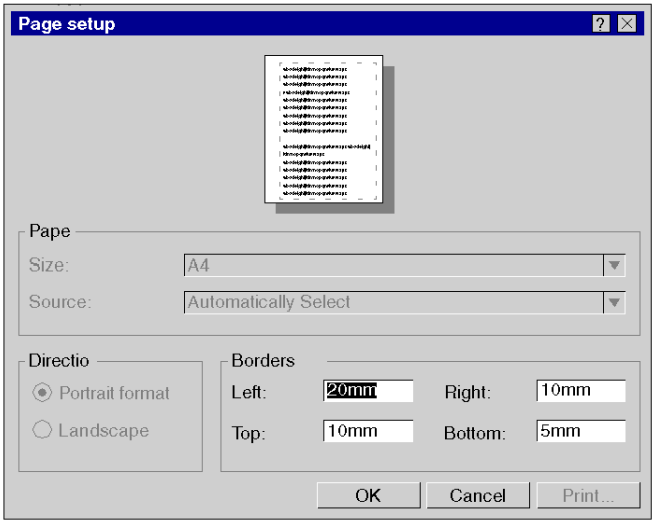
## Introduction

The following options are available to open the setup dialog box:

- In the documentation structure, right-click the subject to include and click **Page Setup...**.
- In the documentation structure, select the subject to include and execute the menu command **Edit → Page Setup...**.

## Representation

Representation of the dialog box:



## Items

Elements of the dialog box:

| Element | Description |
|---|---|
| Preview | Shows the preview for the documentation using the current settings. |
| **Paper** | **Size:**<br>Shows the paper size (the paper format). This setting depends on the current printer and cannot be modified here.<br>**Source:**<br>Shows the paper source. This setting depends on the current printer and cannot be modified here. |
| **Direction** | Shows the orientation of the document on the page. This setting depends on the current printer and cannot be modified here. |
| **Borders** | **Left:**<br>Enter the desired border distance from the left-hand edge of the page.<br>**Right:**<br>Enter the desired border width to the right-hand edge of the page.<br>**Top:**<br>Enter the desired border width to the top edge of the page.<br>**Bottom:**<br>Enter the desired border width to the bottom edge of the page. |
| **OK** | Saves all changes and closes the dialog box.<br>You are not asked if you want to accept the changes, as this requires a new generation of documentation *(see page 1708)*. |
| **Cancel** | Cancels all changes and closes the dialog box. |

# Print

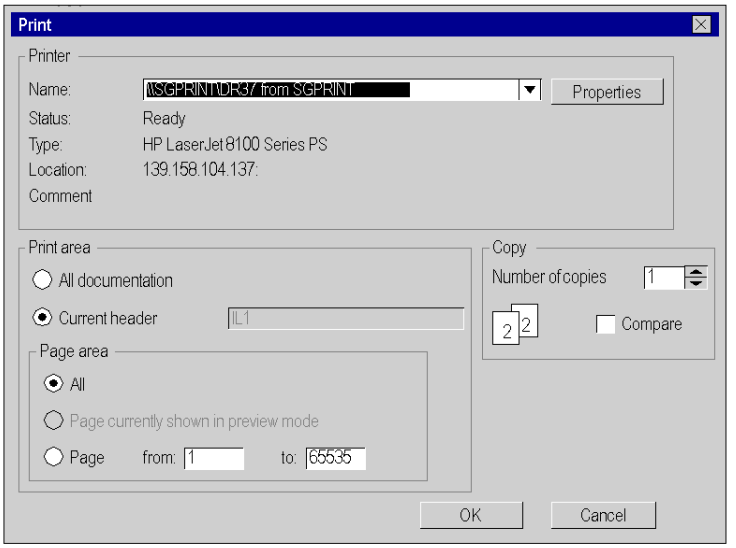
## Introduction

The following options are available to open the printer selection dialog box:

- In the documentation structure, right-click the subject to print and click **Print**.
- In the documentation structure, select the subject to print and execute the menu command **View → Print**.
- In the **print preview**, right-click the preview page and click **Print...**.
- In the documentation structure, select the subject to print and press the key combination **Ctrl**+**P**.

## Representation

Representation of the dialog box:



## Items

Elements of the dialog box:

| Element | Description |
|---|---|
| **Printer** | **Name**<br>Shows the name of the currently selected printer. You can select a different printer from the list box.<br>**Status**<br>Shows the status of the currently selected printer.<br>**Type**<br>Shows the type of the currently selected printer.<br>**Position**<br>Shows the path of the currently selected printer if you are connected to a network printer, or the connection to the computer if you are using a local printer.<br>**Comment**<br>Shows the comment for the currently selected printer.<br>**Properties**<br>Opens the properties dialog box for the currently selected printer. |
| **Print range** | **Whole documentation**<br>If you click this option button, all the subjects *(see page 1707)* in the document structure are printed out.<br>**Current heading**<br>If you click this option button, the currently selected subject and all its sub-sections are printed out. (The header for the current subject is shown in the text box). |
| **Page range** | This area is only available when the option button **Current heading** is activated.<br>**All**<br>If you click this option button, all the pages in the currently selected subject and all its sub-sections are printed out.<br>**Currently previewed page**<br>If you click this option button, the page currently showing in the preview is printed out.<br>(This option is only available if Print preview *(see page 1709)* is active.)<br>**Pages from** / **to**<br>If you click this option button, you can set which pages in the currently selected subject and/or all its sub-sections are printed out. |
| **Copies** | **Number of copies:**<br>Here you can enter the number of copies to print.<br>**Collate**<br>If you check this checkbox, when several copies are printed, each set of document is printed together.<br>If you uncheck this checkbox, when several copies are printed, several copies of each page are printed together. |
| **OK** | Closes the dialog box and starts the printout. |
| **Cancel** | Closes the dialog box without starting the printout. |

# Printout Setup

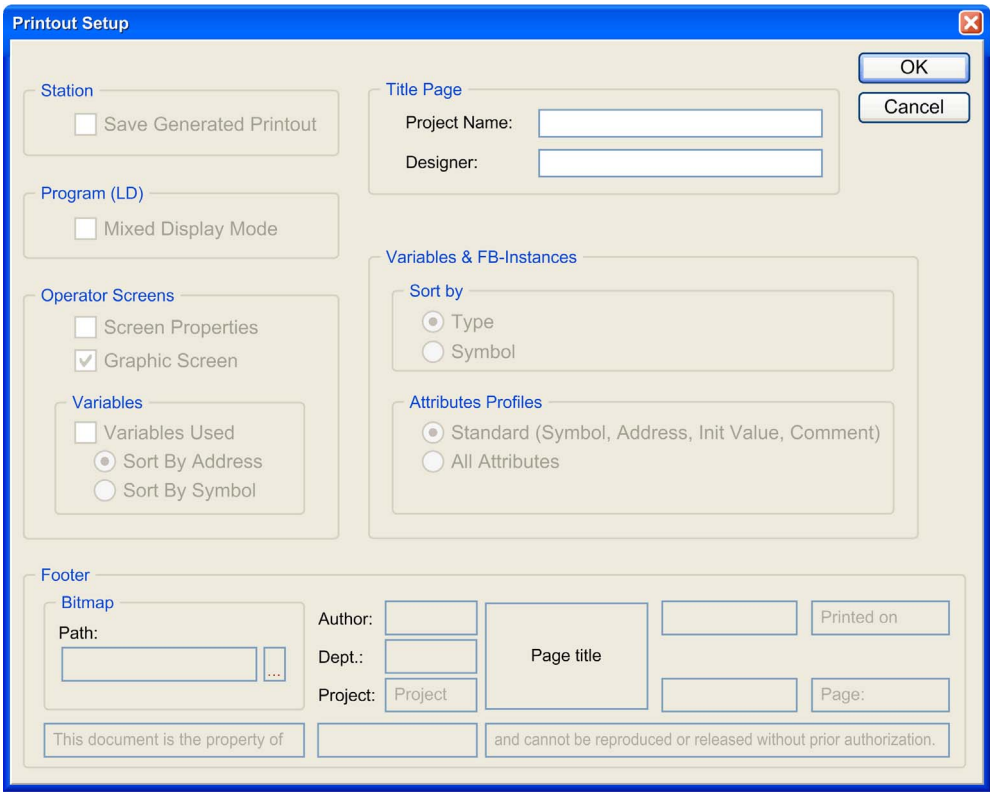
## Introduction

The following options are available to open the printing parameters dialog box:

- Right-click **Documentation → Title page** directory in the project browser and click **Printout Setup...**
- In the documentation structure, right-click a subject and click **Printout Setup...**.
- In the documentation structure, select a subject and execute the menu command **Edit → Printout Setup...**.

## Representation

Representation of the dialog box:



## Items

Elements of the dialog box:

| Element | Description |
|---|---|
| **Station** | **Save Generated Printout**<br>You can save documents that have been prepared *(see page 1708)* between Unity Pro sessions (not saved by default).<br>If you want to save documents, check the checkbox. |
| **Program (LD)** | This area is used to define the display for the documentation for the LD section *(see page 1704)*.<br>**Mixed Display Mode**<br>When you activate this checkbox, the documentation for the LD section is shown in combined display mode *(see page 721)*.<br>When you deactivate this checkbox, the documentation for the LD section is shown in input mode *(see page 720)*. |
| **Operator Screens** | This area is used to define the scope and display for the operating screen documentation *(see page 1705)*.<br>**Screen Properties**<br>When you activate this checkbox, the properties for the operating screens are documented.<br>**Graphic Screen**<br>When you activate this checkbox, the graphical displays for the operating screens are documented.<br>**Variables → Variables Used**<br>When you activate this checkbox, the variables used in the operating screens are documented.<br>**Variables → Sort By Address**<br>If you click this option button, the variables used are sorted according to the address.<br>**Variables → Sort By Symbol**<br>If you click this option button, the variables used are sorted according to the symbol. |
| **Title page** | This area is used to define the title page *(see page 1700)* for the documentation.<br>**Project Name**<br>Enter the name of the project.<br>**Designer**<br>Enter the name of the designer. |

| Element | Description |
|---|---|
| **Variables & FB-Instances** | This area is used to define the scope and display for the variables and FB instances *(see page 1703)*.<br>**Sort by → Type**<br>If you click this option button, the variables and FB instances used are sorted according to type.<br>**Sort by → Icon**<br>If you click this option button, the variables and FB instances used are sorted according to symbol.<br>**Attributes Profiles → Standard (Symbol, Address, Init Value, Comment)**<br>If you click this option button, only the default attributes for the variables and FB instances used are documented.<br>**Attributes Profiles → All Attributes**<br>If you click this option button, all attributes for the variables and FB instances used are documented. |
| **Footer** | This area is used to define the footer *(see page 1706)* for the documentation.<br>**Bitmap → Path:**<br>Here you can enter the path for the bitmap file (for example, logo) to appear in the footer.<br>**Author:**<br>Enter the name of the author.<br>**Dept.:**<br>Enter the name of the department.<br>**Project:**<br>Displays the project name entered in **Title Page**.<br>**Page title**<br>Automatically generated and contains the title of the subject printed.<br>**Free fields:**<br>You can enter any text here.<br>**Printed on**<br>Automatically generated.<br>**Page:**<br>Automatically generated |
| **OK** | Saves all changes and closes the dialog box.<br>**NOTE:** Subjects whose settings have to be changed, need to be generated again *(see page 1708)*. |
| **Cancel** | Cancels all changes and closes the dialog box. |

# Chapter 52
# Import / Export

## Subject of chapter

This chapter describes the procedures for importing or exporting the project program or elements thereof. For example:

- import / export of the inputs / outputs configuration,
- import / export of DFB types,
- import / export of a program section,
- import / export of runtime screens,
- import / export of an animation table, etc.

It is also possible to Import/Export Project settings *(see page 516)* and Import/Export Options *(see page 554)*.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 52.1 | General information on the Import/Export function | 1720 |
| 52.2 | Importing/Exporting the various elements of Unity Pro | 1730 |
| 52.3 | Wizard for importing the various elements of Unity Pro | 1761 |

# Section 52.1
## General information on the Import/Export function

### Object of this sub-chapter

This sub-chapter presents general information on the Import/Export function available under Unity Pro.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| General information on import/export | 1721 |
| Import / Export Files | 1722 |
| Type of dialog box for the export function | 1723 |
| Type of dialog boxes for the import function | 1726 |
| Managing Conflicts | 1728 |

# General information on import/export

## At a Glance

The import / export functions are accessible from the structural view and the functional view of the project browser:

- the import function allows you to retrieve part or all of a project program for use in your project,
- the export function allows you to copy part or all of your project program to a file. You can then retrieve this program for use in a new project using the import function.

## Export function

The export function generates a file that contains **unprotected data** and **references to protected data**. You must define the name of this file and its location (directory), the extension of which is determined by the export type (for example, XDB if you export a DFB type, etc.).

## Import function

When you perform a partial import, the software asks you if you want to save the current project. This allows you to have a pre-import version of the project.

If during an import an element with the same name is found in the project, a dialog box allows you to choose between a number of operating modes:

- **Keep**: the element that is already present in the project is kept (the element with the same name is not imported),
- **Replace**: the element that is already present in the project is replaced by the imported element with the same name,
- **Rename**: you can change the name of the element to be imported in order to remove the conflict.

Which of these three operating modes you choose depends on the element you wish to import.

Following an import, you must confirm the imported data (analysis and generation). In fact, import works like a manual input and **is not confirmed automatically**.

# Import / Export Files

## At a Glance

The files generated during an export contain **unprotected data** and/or **coded protected data**.

The extension of each file is defined by its content, That is to say by your position in the directory tree of the project browser at the time of the export.

## Import / Export Files

The files generated during an export are as follows:

| Element to be exported | File type |
|---|---|
| Inputs/outputs configuration | XHW |
| Program | XPG |
| Section in LD language | XLD |
| Section in IL language | XIL |
| Section in ST language | XST |
| Section in FBD language | XBD |
| Section in SFC language | XSF |
| Segment in LL984 language | X9S |
| Network in LL984 language | X9N |
| DFB type | XDB |
| DDT | XDD |
| Variables | SCY / TXT / XSY / XVM |
| Communication network(s) | XCM |
| Runtime screen(s) | XCR |
| Animation table | XTB |
| Functional module | XFM |
| Global project | XEF |
| Global project with global DTM configuration | ZEF |

**NOTE:** When the first letter of the file extension is ’X’, this means that the file in XML format (extensible markup language).

# Type of dialog box for the export function

## At a Glance

When you select the export function from the Unity Pro menu or from an element to be exported, two types of dialog boxes appear on the screen. These dialog boxes allow you to define:

- The name of the export file,
- The location of the file,
- The type of file,
- The export options.

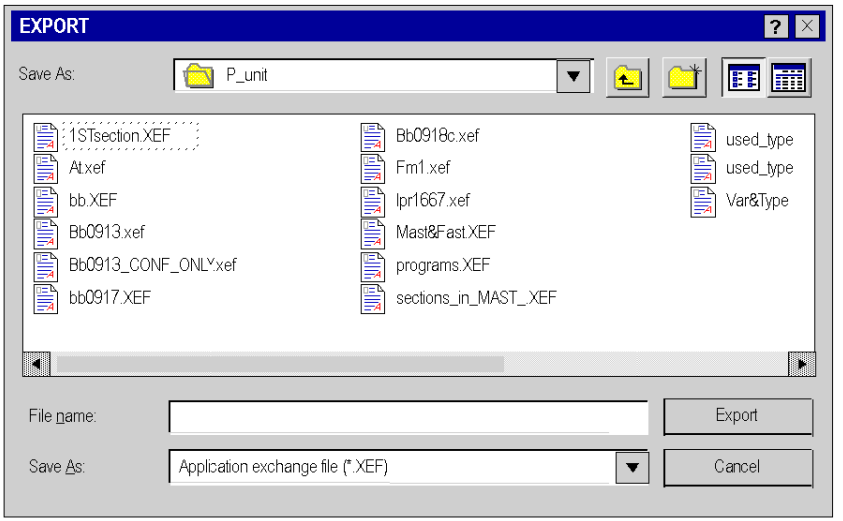
## Standard dialog box

The following dialog box appears when you export one of the following element types:

- Inputs/outputs configuration *(see page 1731)*,
- DFB type *(see page 1737)*,
- DDTs *(see page 1739)*,
- Communication network *(see page 1745)*,
- Operator screen *(see page 1747)*,

**NOTE:** For all these elements to be exported, simply define the name of the export file and its location. The type of file under which the export is saved cannot be modified.

The following example shows the dialog box for exporting a global project:



## Dialog box and options

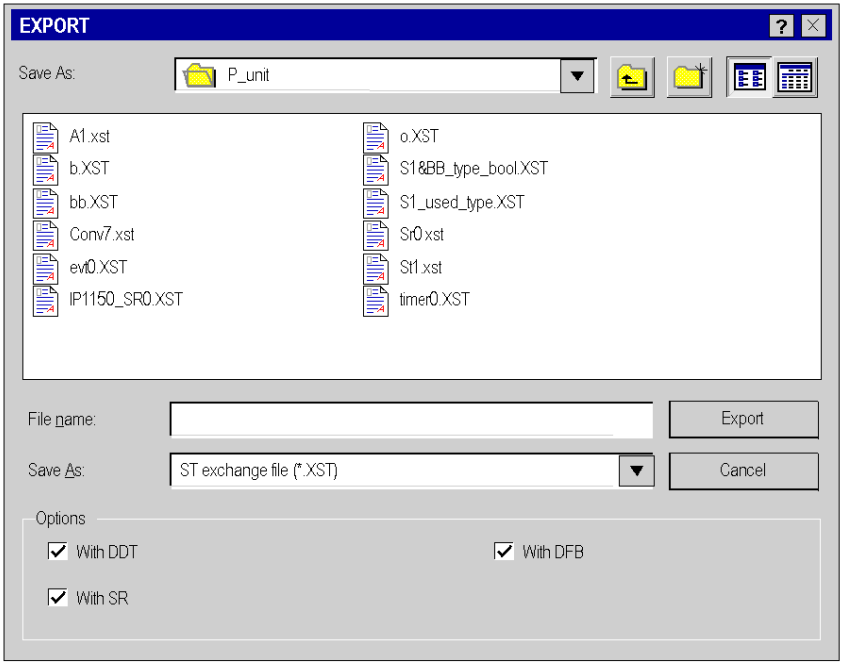
To export the following elements, you must define the name of the file with its location. You can choose 1,2 or 3 options:

- Program and section *(see page 1733)*,
- Variables *(see page 1741)*,
- Animation table *(see page 1750)*,
- Functional module *(see page 1752)*.
- Global project *(see page 1755)*.

The following table describes the available options as a function of the elements to be exported:

| Type of element to be exported | Choice of file type | Export mode with or without DDT option | Export mode with or without DFB type option | Export mode with or without SR option | Export mode with or without conf option |
|---|---|---|---|---|---|
| Program | no | yes | yes | yes | no |
| Section | no | yes | yes | yes | no |
| Variables | yes | yes | yes | no | no |
| Animation table | no | yes | yes | no | no |
| Functional module | no | yes | yes | no | no |
| Global project | no | no | no | no | yes |

The following example shows the export dialog box for a section element, including the checkbox options:



# Type of dialog boxes for the import function

## At a Glance

When you select the import function from the Unity Pro menu or from an element to be imported, various dialog boxes appear on the screen. These dialog boxes allow you to define:

- The name of the import file,
- The location of the file,
- The type of file,
- The import option,
- The import procedures (Keep, Replace, Rename).

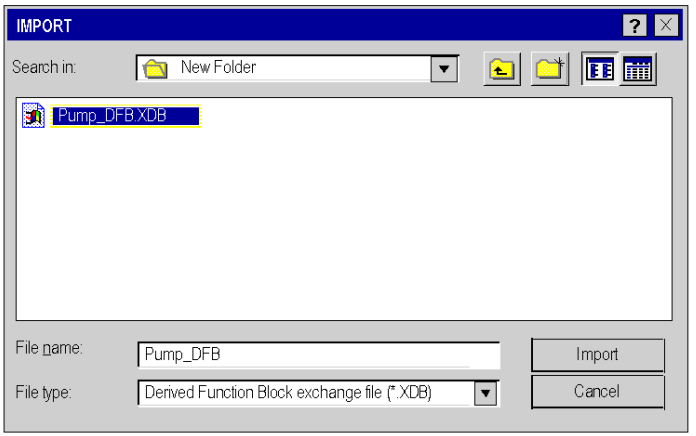
## Standard dialog box

The following dialog box appears when you import one of the following element types:

- Inputs/outputs configuration *(see page 1732)*,
- DFB type *(see page 1738)*,
- DDTs *(see page 1740)*,
- Variables *(see page 1744)*,
- Communication network *(see page 1746)*,
- Animation tables *(see page 1751)*,
- Operator screens *(see page 1749)*.

**NOTE:** For all these elements to be imported, simply define the name of the import file and its location. The type of file under which the import is chosen cannot be modified except for importing variables.

The following example describes the dialog box for importing a DFB type:



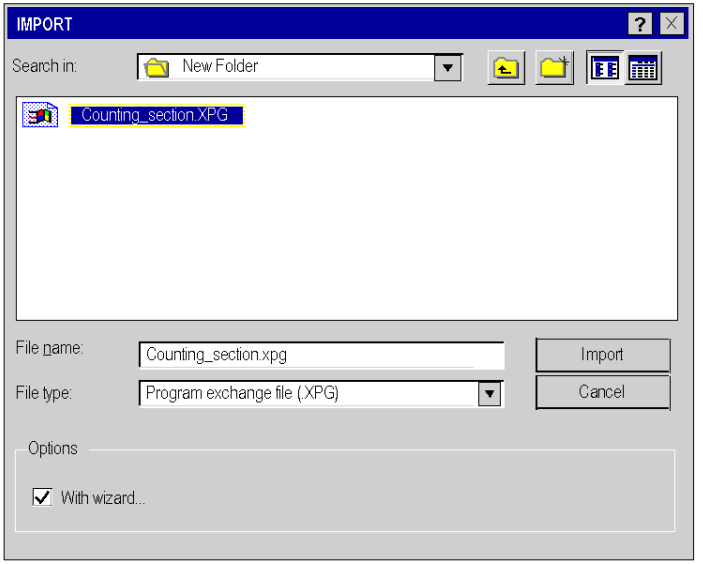
## Dialog box with wizard

In order to facilitate reassignments, you can choose to import the following elements using wizard *(see page 1762)*:

- Global project *(see page 1757)*,
- Sections *(see page 1735)*,
- Functional modules *(see page 1754)*.

**NOTE:** For all these elements to be imported, simply define the name of the import file and its location and check the option **With wizard**.

The following example shows the dialog box for importing a section:



# Managing Conflicts

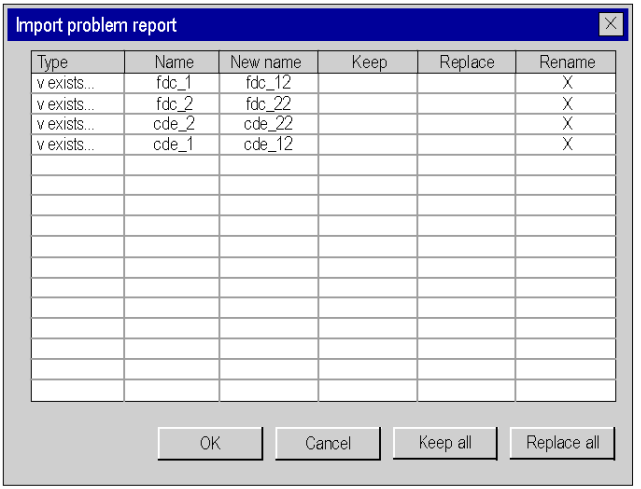
## At a Glance

When importing an element with the same name, if it already exists in the project, an **Import problem report** dialog box lets you choose from several operating modes *(see page 1721)*:

- Keep,
- Replace,
- Rename.

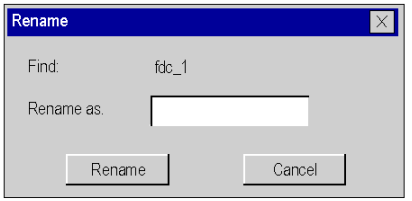
These operating modes depend on the element to be imported. Excluding the global project import function, all the other elements have these operating modes.

The following example shows the dialog box with conflicts when importing variables:



## How to Manage Conflicts

You should perform the following steps in order to import an element and manage conflicts during the import process:

| Step | Action |
|---|---|
| 1 | In the project browser, select the element to be imported.<br>**Result**: the element selected appears in reverse video. |
| 2 | Activate the **Import** command from the **File** menu or from the contextual menu (by clicking the right mouse button).<br>**Result**: a dialog box appears on the screen: |
| 3 | Choose the source folder for the import (directory tree) then select the file to be imported. For the **Variables** element, the file type can be modified (XSY or TXT).<br>**Result**: the name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: if there is no conflict, a progress indicator appears during the import, otherwise the **Import problem report** dialog box appears on the screen. |
| 5 | If you do not want to import the elements, check the **Keep** mode. |
| 6 | If you want to replace the elements with the imported ones, check the **Replace** mode. For further information on the replacement of Function Block (FB) instances see *Changing the type of an instance, page 366*. |
| 7 | If you want to rename the elements, check the **Rename** mode.<br>**Result**: a dialog box appears on the screen where you can enter the name for the element to be modified.<br>  |
| 8 | Start the import with your chosen option (Keep, Replace, Rename) using the **OK** command. If a conflict still exists, the **Import problem report** appears on the screen. Repeat steps 5, 6 and 7.<br>If you want to cancel the import, activate the **Cancel** command. |

# Section 52.2
## Importing/Exporting the various elements of Unity Pro

### Object of this sub-chapter

This sub-chapter presents the Import/Export functions that apply to the various elements of Unity Pro.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Export of the input/output configuration | 1731 |
| Import of the input/output configuration | 1732 |
| Exporting the program | 1733 |
| Program Import | 1735 |
| Exporting DFB types | 1737 |
| Importing DFB types | 1738 |
| Exporting DDTs | 1739 |
| Importing DDTs | 1740 |
| Exporting Variables | 1741 |
| Importing Variables | 1744 |
| Exporting the communication network configuration | 1745 |
| Importing the configuration of a communication network | 1746 |
| Exporting operator screens | 1747 |
| Importing operator screens | 1749 |
| Exporting animation tables | 1750 |
| Importing animation tables | 1751 |
| Exporting a functional module | 1752 |
| Importing a functional module | 1754 |
| Exporting a global project | 1755 |
| Importing a global project | 1757 |
| Exporting/Importing LL984 Segments/Networks | 1759 |

# Export of the input/output configuration

## At a Glance

This function can be accessed from the **Configuration** folder in the project browser. It allows you to export the complete configuration of the inputs/outputs of an existing project, that is to say its physical description, in order to be able to use it in a new project. You can, for example, export:

- the bus parameters,
- the racks,
- the modules,
- the channel parameters.

## File generated

When you export the input/output configuration you copy **the entire configuration** of the project to a **.XHW** file. This file is in XML format.

**NOTE:** On Modicon M580 platform, **.XHW** export file does not include the DTM configuration part of the CPU. Any configuration parameter from the DTM (eg : Device list, Channel properties, ..etc) will not be included in the **.XHW** export file.

## Exporting the input/output configuration

To export the input/output configuration, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the project browser, select the **Configuration** folder.<br>**Result**: The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or from the context-sensitive menu (accessible with a right-click).<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the name of the file. |
| 4 | Activate the **Export** command.<br>**Result**: a progress indicator lets you know how the export is proceeding. |
| 5 | A message in the **output window** tells you that export is complete. |

# Import of the input/output configuration

## At a Glance

This function is accessible from the **Configuration** folder in the project browser. It allows you to import a complete input/output configuration. If you have already configured your project (for example if you have modified the default configuration), the software offers you a choice between:

- replacing your current input/output configuration with the one contained in the import file,
- canceling the import in order to keep your current input/output configuration.

## Importing the input/output configuration

To import an input/output configuration, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the project browser, select the **Configuration** folder.<br>**Result**: This folder appears in reverse video. |
| 2 | Activate the **Import** command from the **File** menu or from the contextual menu (accessible with a right-click).<br>**Result**: A dialog box appears on the screen. |
| 3 | Choose the source directory for the import (directory tree), then select the **.XHW** file to be imported.<br>**Result**: The name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command. |
| 5 | A message tells you that the import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

**NOTE:**
When importing a configuration with Device DDT instance:

- managed Device DDT instances become unmanaged
  After importing the configuration the user have to link manually all the Unmanaged Device DDT instances with each device.
- EIO modules are inserted with default Device DDT

**NOTE:** A new Device DDT instance added can possibly match with an unmanaged Device DDT instance.

# Exporting the program

## At a Glance

This function is accessible from the project browser.

Depending on the element that you have selected in the directory tree of the project browser, it allows you to export:

- either all the sections and program modules (SR) of a sequential task (MAST, FAST, AUX),
- or a section of a sequential task (excluding the section activation conditions),
- or an SR program module of a sequential task,
- or **Timer** or **Input/Output** event processing.

**NOTE:** When you export an SFC section, the sections containing the actions and transitions (whether referenced or not) are also exported.

**NOTE:** When you export a conditioned section, the conditions are not exported. They are exported only if you export the complete task.

## Options associated with exporting the program

Depending on the program elements you are exporting, you have a choice of 3 options:

- export with or without the content of all the DDTs used in the program element,
- export with or without the content of all the DFB types used in the program element,
- export with or without the content of all the SR program modules called in the program element.

These 3 **Options** are selected by default.

## Files generated

When you export a set of sections and/or program modules or event processing elements, the software generates a **.XPG** file which contains the following **unprotected information**: sections, SR program modules, event processing elements, unprotected DFB types used (if the option is activated), DDTs used (if the option is activated).

When you export a section or SR program module or an event processing element, the software generates a file according to the programming language used:

- **.XLD**, if the language of the element exported is LD
- **.XIL**, if the language of the element exported is IL
- **.XST**, if the language of the element exported is ST
- **.XSF**, if the language of the element exported is SFC (sections only)
- **.XBD**, if the language of the element exported is FBD

The file contains the **unprotected information**: program section, module or current event processing, unprotected DFB types used (if the option is activated), DDTs used (if the option is activated), program modules called (if the option is activated).

**NOTE:** The sections protected (read/write) by the active project protection *(see page 186)* cannot be exported.

## Exporting a program from the project browser

To export a program from the project browser, perform the following operations:

| Step | Action |
|---|---|
| 1 | Select the element to be exported.<br>**Result**: The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the file name. |
| 4 | Select export mode: with or without DDT, DFB types, SR called.<br>**Result**: when a **With** option is selected, the corresponding box is checked. |
| 5 | Activate the **Export** command.<br>**Result**: a progress indicator lets you know how the export is proceeding. |
| 6 | A message in the **output window** tells you that export is complete. |

# Program Import

## At a Glance

This function Is accessible from the Project Browser. According to the element you selected in the project browser tree, you can use it to import:

- Either all of the sections and SR program modules of a sequential task (MAST, FAST, AUX). The sections and program modules are created in the same order as they were when memorized during export.
- One or all of the sections of a sequential task. These sections are imported after the sections that already exist in the list.
  If you need to perform any reassignments, a wizard is available (except for protected sections which cannot be modified).
- One or all of the program modules of a sequential task. These modules are imported after the modules that already exist in the list.
- One or all of the **Timer** or **Input/Output** event processing operations. These processing operations are imported after the event processing operations that already exist in the list.

If one of the previous elements already exists in the current project, the software asks you to choose between the following options:

- **keep** the component of the current project,
- **replace** the project component by the one from the import file,
- **rename** the imported element.

**NOTE:** When importing a program element, if you wish to **keep** an element which has the same name in the project (program), then you need to rename the imported element (the name of an element is unique in a project).

**NOTE:** If the DDT, DFB types, and SR known as associated were exported (options selected during export), they are imported with the program element.

**NOTE:** After importing the section or program module, validate them.

## How to Import a Program Element

Perform the following operations to import a program element:

| Step | Action |
|---|---|
| 1 | In the project browser, select the folder into which you wish to import you program element (sections, SR or events).<br>**Result**: this folder appears in reverse video. |
| 2 | Activate the **Import** command from the **File** menu or from the contextual menu.<br>**Result**: a dialog box appears on the screen: |
| 3 | Choose the source folder for the import (directory tree) then select the file to be imported (XPG, XLD, XIL, XST, and so on).<br>**Result**: the name of the file then appears in the **File name** field. |
| 4 | Select the **With wizard** option if you would like help in reassigning elements. |
| 5 | Activate the **Import** command.<br>**Result**: a progress indicator shows how the import is progressing. |
| 6 | A message tells you when the import is complete.<br>If errors occur during the import, they are indicated by a message in the **display window**. |

# Exporting DFB types

## At a Glance

This function is accessible via the **Derived FB types** folder in the project browser.

Depending on the element that you have selected in the directory tree of the project browser, it allows you to export:

- either all the DFB types of the current project (even those that have not been used),
- or one DFB type.

**NOTE:** if you export a DFB type, which uses nested DFBs or DDTs, these are also exported automatically.

## Protected DFB.

A **protected** DFB type (in write or read/mode) can be exported and keeps the same type of protection.

## File generated during export of all DFBs

When you export all the DFB types of the project, the software generates an **.XDB** file containing

- all the **unprotected information**: unprotected nested DFB types, DDTs used, and references to the protected DFB types

## File generated during export of one DFB type

When you export one DFB type, the software generates an**.XDB** file containing

- all the **unprotected information**: current DFB type, nested DFB types, DDTs used, and references to the protected DFB types.

## Exporting one or more DFB type(s) from the project browser

To export one DFB type or all the DFB types of a project, perform the following operations from the project browser:

| Step | Action |
|---|---|
| 1 | Select the element to be exported.<br>**Result**: The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the file name. |
| 4 | Activate the **Export** command.<br>**Result**: a progress indicator lets you know how the export is proceeding. |
| 5 | A message in the **output window** tells you that export is complete. |

# Importing DFB types

## At a Glance

This function is accessible from the **Derived FB types** folder in the project browser. It allows you to import one or several DFB types.

**NOTE:** A protected DFB type can be imported. After import if it is write protected, the elements of the DFB will be accessible in read only. If it is read/write protected, the elements are not accessible. If it is version protection protected, the version number is accessible in read only and the elements of the DFB are accessible in read/write.
If the DFB type uses nested DFBs or DDTs, these are also imported.

If a DFB type already exists in the current project, the software proposes the following options:

- **keep** the element from the current project,
- **replace** the project element with the one in the import file,
- **rename** the imported element.

## Importing a DFB type

To import a DFB type, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the project browser, select the **Derived FB Types** folder.<br>**Result**: This folder appears in reverse video |
| 2 | Activate the **Import** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Choose the source directory for the import (directory tree), then select the file to be imported (XDB).<br>**Result**: The name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: A progress indicator lets you know how the import is proceeding. |
| 5 | A message tells you that the import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

# Exporting DDTs

## At a Glance

This function is accessible from the **Derived Data Types** folder in the project browser.

Depending on the element that you have selected in the directory tree of the project browser, it allows you to export:

- All the current project DDTs (even those that have not been used).
- One DDT.

**NOTE:** If you export a DDT that uses nested DDTs, these are also exported automatically.

## File generated during export of all DDTs

When you export all the DDTs, the software generates an **.XDD** file which contains all the **unprotected information** for the project DDTs.

## File generated during export of one DDT

When you export one DDT, the software generates an **.XDD** file which contains all the **unprotected information** such as the current DDT and DDTs used.

## Exporting one or more DDTs from the project browser

To export one DDT, or all the DDTs, of a project, perform the following operations from the project browser:

| Step | Action |
|---|---|
| 1 | Select the element to be exported.<br>**Result**: The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the file name. |
| 4 | Activate the **Export** command.<br>**Result**: a progress indicator lets you know how the export is proceeding. |
| 5 | A message in the **output window** tells you that export is complete. |

# Importing DDTs

## At a Glance

This function is accessible from the **Derived Data Types** folder in the project browser. It allows you to import one or several DDTs.

**NOTE:** If the DDT uses nested DDTs, these are also imported.

If a DDT type already exists in the current project, the software proposes the following options:

- **keep** the element from the current project.
- **replace** the project element with the one in the import file.
- **rename** the imported element.

## Importing a DDT

To import a DDT, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the project browser, select the **Derived Data Types** folder.<br>**Result**: This folder appears in reverse video. |
| 2 | Activate the **Import** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Choose the source directory for the import (directory tree), then select the **.XDD** file to be imported.<br>**Result**: The name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: A progress indicator lets you know how the import is proceeding. |
| 5 | A message tells you that the import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

# Exporting Variables

## At a Glance

This function is accessible from the **FB Variables and instances** folder in the project browser.

According to the element you selected in the project browser tree, you can use it to export:

- all the variables of the project (even those that are not used),
- all the variables of a given family: EDT, IODDT, and so on (even those that are not used).

To export a subset of the variables, see also the sub-section Export of a subset of variables *(see page 416)*.

## Excel Tool for Variables Management

An Excel template file is available in Unity Pro extra tools to provide parameters management from files generated in *.XSY* format. The Excel features allow to create, edit, sort, or filter Unity Pro application variables, and import them back in Unity Pro.

The Excel template and its manual are installed with Unity Pro: **Start** → **All Programs** → **Schneider Electric** → **SoCollaborative** → **Unity Pro** → **Extras** → **Excel Import Export Tool**. They are also available in the Unity Pro software DVD (**Extras** → **Excel Import Export Tool**).

## Format of Variable Files

Variables can be saved:

- either in source format, in an *.SCY* file compatible with the PL7 program,
- or in text format with tab separators, in a *.TXT* file,
- or in XML format, in an OFS-compatible *.XVM* file,
- or in XML format, in an *.XSY* file.

## Options Associated with Exporting Variables

Whatever the variables you export, you can choose two options:

- export with or without the content of all the DDTs used,
- export with or without the content of all the DFB types used.

By default, the two "export with" options are selected.

**NOTE:** These two options are available only when exporting files to *.XSY* format.

## Purpose of Importing Data

Importing data into a specific format as *.XSY*, *.SCY*, *.TXT* or *.XVM* can be useful for different purposes.

Those functionalities are described in the table below:

| File type | Available for |
|---|---|
| *.XSY* | Export / Import |
| *.SCY* | Export / PL7 |
| *.TXT* | Export / Import / Excel |
| *.XVM* | Export / OFS |

## Files Structure For .XSY, .SCY, .TXT And .XVM Formats

When importing elementary variables from the data editor screen, the data structure with extension *.XSY*, *.SCY*, *.TXT*, and *.XVM* is displayed in the following order: *Name*, *Address*, *Type*, *Comment*, and *Value*.

**NOTE:** The structure of imported data from a *.XSY* file can have additional information. If the options are set in column configuration screen, the complete structure for this type of file is the following: *Name*, *Address*, *Type*, *Comment*, *Value*, *Save*, *RW Program*, *Constant* and *Custom*.

## Files Generated In .XSY And .XVM Formats

When you export variables, the program generates an *.XSY* or an *.XVM* file containing the unprotected information, such as the list of current variables, the unprotected DFB types of which an instance is exported (if the option is set), and the DDTs of which an instance is exported (if the option is set).

## Files Generated In .SCY, .XVM, .TXT Formats

When you export variables, the program generates a *.TXT* or an *.SCY* file containing the unprotected information about current variables.

If you want to use Microsoft Excel to open the exported *.TXT* file, please follow the steps:

- Open Microsoft Excel
- With **Open** command of Excel, open the **.TXT** file by selecting in the open window **Text files** as type of file.
- When the Excel file import assistant window opens, select **Delimited** for the type of file. Click on the **Next** button.
- In the next window, check **Tab** option box as column separator. Click on the **Next** button.
- In the next window select in the display the column that contain the values of the variables and click on the **Text** option button for the column data format. Doing so keeps the same values as in the *.TXT* file. If this format is not used for the column, positive values will appear as negative when opening the *.TXT* file in Excel.

**NOTE:** The variables must not exceed 32 characters in length. Once you have modified the *.TXT* file, save it with the tab-separated value option before importing.

## How to Export a List of Variables from the Project Browser

Perform the following operations to export all the variables or a family of variables from the project browser:

| Step | Action |
|---|---|
| 1 | Select the element to be exported in the **FB Variables and instances** folder.<br>**Result:** The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or from the contextual menu.<br>**Result:** A dialog box appears on the screen: |
| 3 | Select the target folder for the export (directory tree) then enter the file name. |
| 4 | Select the export mode. with or without DDT, DFB type.<br>**Result:** When a "with" option is selected, the corresponding box is checked. |
| 5 | Activate the **Export** command.<br>**Result:** A progress indicator shows how the export is progressing. |
| 6 | A message is displayed in the **display window** to indicate that the export is finished. |

# Importing Variables

## At a Glance

This function is accessible from the **FB Variables and instances** folder in the project browser. It enables you to import a list of variables into your project.

**NOTE:** If the DDTs or associated DFB types have been exported (options selected on export), they are imported with the list of variables.

If the list of variables already exists in the current project, the software asks you to choose between the following options:

- **keep** the component of the current project,
- **replace** the project component by the one from the import file,
- **rename** the imported component, allowing you to keep both components.

## Excel Tool for Variables Management

An Excel template file is available in Unity Pro extra tools to provide parameters management from files generated in *.XSY* format. The Excel features allow to create, edit, sort, or filter Unity Pro application variables, and import them back in Unity Pro.

The Excel template and its manual are installed with Unity Pro: **Start** → **All Programs** → **Schneider Electric** → **SoCollaborative** → **Unity Pro** → **Extras** → **Excel Import Export Tool**. They are also available in the Unity Pro software DVD (**Extras** → **Excel Import Export Tool**).

## How to Import a List of Variables

Perform the following operations to import a list of variables:

| Step | Action |
|---|---|
| 1 | In the project browser, select the **FB Variables and instances** folder.<br>**Result**: the **Variables** folder appears in reverse video. |
| 2 | Activate the **Import** command from the **File** menu or from the contextual menu (by clicking the right mouse button).<br>**Result**: a dialog box appears on the screen: |
| 3 | Choose the source folder for the import (directory tree) then select the file to be imported (XSY or TXT).<br>**Result**: the name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: a progress indicator shows how the import is progressing. |
| 5 | A message tells you when the import is complete.<br>If errors occur during the import, they are indicated by a message in the **display window**. |

**NOTE:** You can only import **.XSY** or **.TXT** files. Files in **XVM** and **SCY** format cannot be imported.

# Exporting the communication network configuration

## Introduction

This function is accessible from the **Communication** folder of the project browser.

Depending on the element selected in the directory tree of the project browser, it allows you to export:

- All communication networks with their configurations.
- A specific communication network with its configuration.

## File generated during export of networks

When you export one or all of the communication networks, the software generates a **.XCM** file containing all the information on each network: common data, bridge.

## How to export an operator screen

To export the communication networks, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the **Project** browser, select the element to be exported.<br>**Result**: The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the file name. |
| 4 | Activate the **Export** command.<br>**Result**: A progress indicator lets you know how the export is proceeding. |
| 5 | A message in the **output window** tells you that export is complete. |

**NOTE:** If you check the **With Conf.** option box before the export, the networks are exported with the links to the modules of the project. When you import the networks in another project they are automatically linked to the existing modules. This option works only if the modules of the other project are compatible with those of the previous one and if they are placed in the same positions in the rack.

# Importing the configuration of a communication network

## At a Glance

This function is accessible from the **Communication** folder of the project browser.

You can import one or more communication network configurations.

If a communication network already exists in the current project with the same name, the software proposes the following options:

- **keep** the element from the current project.
- **replace** the project element with the one in the import file.
- **rename** the imported element.

## How to import a functional module

To import a communication network, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the function view of the project browser, select the **Communication** folder.<br>**Result**: This folder appears in reverse video. |
| 2 | Activate the **Import** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Choose the source directory for the import (directory tree), then select the file to be imported (XCM).<br>**Result**: The name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: A progress indicator lets you know how the import is proceeding. |
| 5 | A message tells you that the import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

# Exporting operator screens

## Introduction

This function is accessible from the **Operator Screens** folder in the project browser.

Depending on the element that you have selected in the directory tree of the project browser, it allows you to export:

- All the operator screens of the project.
- All operator screens in the same family.
- A specific operator screen.

**NOTE:** The messages associated with the screens are only exported if you export all the operator screens *(see page 1748)* of the project.

## Files generated during the export of an operator screen

When you export one or several operator screens, the software generates an **.XCR** file for each family of screens. This file contains all the information relating to the graphic objects and the reference of the image objects (.bmp, .jpg). The image object references correspond to their names and their access paths.

## How to export an operator screen

To export an operator screen (all screens, a family of screens or a single screen), perform the following operations:

| Step | Action |
|---|---|
| 1 | In the **Project** browser, select the element to be exported.<br>**Result**: The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the file name. |
| 4 | Activate the **Export** command.<br>**Result**: A progress indicator lets you know how the export is proceeding. |
| 5 | A message in the **output window** tells you that export is complete. |

## Import/Export of messages

The messages associated with the screens are only exported if you export all the operator screens of the project. If you export a single operator screen or an operator screen family, the messages are not exported.

That means when you export a single screen from project A and import it to project B, you need to redefine all messages used inside this screen in the **Messages list** of Project B.

As a workaround you can export and import all messages as described in Messages editing and import/export function *(see page 1664)*.

Another way to use messages created in project A in screens imported to project B, is to open project A and project B in parallel and to copy/paste the messages from the **Messages list** in project A to the **Messages list** in project B.

You have to take care that the **Message numbers** meet the demands of your new project.

# Importing operator screens

## At a Glance

This function is accessible from the **Operator Screens** folder in the project browser.

In the **Operator Screens** folder, if you are importing one or several screens with their families you can:

- import a family of screens, in which case it is then inserted in the folder. The screens belonging to this family continue to be located in their families of origin.
- import the screens without families directly in the root of the folder (they do not belong to a family).

**NOTE:** If you import a file containing references to images and the access path to these images is no longer correct, the images do not appear in the imported screen.

**NOTE:** Screens are only reattached to the functional modules to which they were attached before export when you import a complete ZEF, XEF or XFM file. If you import an operator screen in the structural view (XCR file), the attachment between operator screen and functional module is lost. If you import an operator screen in the functional view (XCR file), it is attached to the functional module from which the import was launched.

## Import/Export of messages

Please refer to *Import/Export of messages* *(see page 1748)*.

## Importing an operator screen

To import an operator screen, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the project browser, select the **Operator Screens** folder into which you wish to import your screen or family of screens.<br>**Result**: This folder appears in reverse video. |
| 2 | Activate the **Import** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Choose the source directory for the import (directory tree), then select the file to be imported (.XCR).<br>**Result**: The name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: A progress indicator lets you know how the import is proceeding. |
| 5 | A message tells you that the import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

# Exporting animation tables

## At a Glance

This function is accessible from the **Animation Tables** folder of the project browser.

Depending on the element that you have selected in the directory tree of the project browser, it will allow you to export:

- all the animation tables of the project,
- a single animation table.

## File generated during export of a single animation table

When you export an animation table, the software generates a **.XTB** file that contains all the **unprotected information** such as the current animation table.

## Exporting a single animation table

To export one or all of the animation tables, perform the following operations from the project browser:

| Step | Action |
|---|---|
| 1 | Select the element to be exported.<br>**Result**: The selected element appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or the contextual menu.<br>**Result**: a dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the file name. |
| 4 | Activate the **Export** command.<br>**Result**: a progress indicator lets you know how the export is proceeding. |
| 5 | A message in the **output window** tells you that export is complete. |

# Importing animation tables

## At a Glance

This function is accessible from the **Animation Tables** folder of the project browser. It allows you to import one or more animation tables into your project.

If an animation table already exists in the current project with the same name, the software proposes the following options:

- **keep** the element from the current project,
- **replace** the project element with the one in the import file,
- **rename** the imported element.

## Importing a single animation table

To import a single animation table, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the project browser select the **Animation Tables** folder.<br>**Result**: This folder appears in reverse video |
| 2 | Activate the **Import** command from the **File** menu or the contextual menu.<br>**Result**: a dialog box appears on the screen. |
| 3 | Choose the source directory for the import (directory tree), then select the file to be imported (XTB).<br>**Result**: The name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: A progress indicator lets you know how the import is proceeding. |
| 5 | A message tells you that import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

# Exporting a functional module

## At a Glance

This function is accessible from the **function view** of the project browser.

It allows you to export all the elements from the functional module selected:

- functional module properties (name, comment),
- sections (with the variables used, program code, activation condition),
- event processing,
- functional sub-modules (nested functional modules),
- operator screens,
- animation tables.

## Options associated with exporting a functional module

As with the program export procedure, you have the choice of 3 options:

- export with or without definition of all the DDTs used in the functional module,
- export with or without definition of all the DFB types used in the functional module,
- export with or without the content of all the SR program modules called in the program element.

The 3 "export with" options are selected by default.

## Files generated

When you generate a functional module, the software generates a **.XFM** file. This contains all the **unprotected information** such as the sections, event processing, DFB types used (if the option has been activated), DDTs used (if the option has been activated) and the SR program modules called (if the option has been activated). It also contains the references of the operator screen images.

## How to export a functional module

To export a functional module, perform the following operations from the function view of the project browser:

| Step | Action |
|---|---|
| 1 | Select the functional module to be exported.<br>**Result**: the functional module selected appears in reverse video. |
| 2 | Activate the **Export** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the destination directory for the export (directory tree) and enter the file name. |
| 4 | Select export mode: with or without DDT, DFB types called.<br>**Result**: when a "with" option is selected, the corresponding box is checked. |
| 5 | Activate the **Export** command.<br>**Result**: a progress indicator lets you know how the export is proceeding. |
| 6 | A message in the **output window** tells you that export is complete. |

# Importing a functional module

## At a Glance

This function is accessible from the **functional view** of the project browser.

You can import a functional module under:

- the station,
- in the existing module, the new module is then a nested module.

In all cases, the assistant is offered as an option.

**NOTE:** Screens are only re-attached to the functional modules to which they were attached before export when you import a complete ZEF, XEF or XFM file. If you import an operator screen in the structural view (XCR file), the attachment between operator screen and funtional module is lost. If you import an operator screen in the functional view (XCR file), it is attached to the functional module from which the import was launched.

## How to import a functional module

To import a functional module, perform the following operations:

| Step | Action |
|---|---|
| 1 | In the function view of the project browser, select the documentation file in which you wish to import your functional module (station or a functional module).<br>**Result**: This folder appears in reverse video |
| 2 | Activate the **Import** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Choose the source directory for the import (directory tree), then select the file to be imported (XFM).<br>**Result**: The name of the file then appears in the **File name** field. |
| 4 | Activate the **Import** command.<br>**Result**: A progress indicator lets you know how the import is proceeding. |
| 5 | A message tells you that import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

# Exporting a global project

## At a Glance

This function is accessible from the **Station** folder of the project browser. It allows you to export the following elements of the project:

- all the DDTs
- all the variables (used and unused)
- all the unprotected DFB types
- the entire program: sections, SR program modules (with execution condition), event processing
- all the functional modules
- all the operator screens
- all the animation tables
- the project properties

This function optionally allows to export the following elements of the project:

- the input/output configuration
- the communication network configuration
- the global DTMs configuration (DTMs topology tree and device configurations)

**NOTE:** The EFs, EFB types, protected DFB types, documentation and user preferences are not exported.

## Files generated

Two formats of export files are available:

- **.XEF** file to export a global project without global DTMs configuration
- **.ZEF** file to export a global project with global DTMs configuration

The export file contains all the **unprotected information**: configuration of the inputs/outputs, sections, SR program modules, events processing, unprotected DFB types, DDTs, variables, animation tables, references to the protected DFB types, global DTMs configuration (ZEF file only).

When exporting protected DFBs, private variables and sections of the DFB are encrypted. However, the variables of type IN, OUT, IN/OUT and PUBLIC still visible.

**NOTE: .XEF** or **.ZEF** file do not contain the configuration of the TSX CPP 110. Do not forget to backup the **.CO** file (Sycon) for CANopen after an export of the project.

## Maintain Output Links on Disabled EF

For compatibility of the project setting **Maintain output links on disabled EF (EN=0)** please refer to the respective section *(see page 542)*.

## How to export a global project

To export a global project, perform the following operations from the project browser:

| Step | Action |
|---|---|
| 1 | Select the **Station** folder.<br>**Result**: This folder appears in reverse video |
| 2 | Activate the **Export project** command from the **File** menu or the contextual menu.<br>**Result**: A dialog box appears on the screen. |
| 3 | Select the export format from the **Type** option:<br>● **.ZEF** format to export a global project with global DTMs configuration<br>● **.XEF** format to export a global project without global DTMs configuration. If the application contains DTMs, the global DTMs configuration is not exported. |
| 4 | Select the destination directory for the export (directory tree) and enter the file name. |
| 5 | If the XEF export format is chosen, select the option **With Conf** to include all data relating to the hardware configuration in the file.<br>**NOTE:** All managed Device DDT become unmanaged if the **With Conf** option is unchecked. The user will link manually all the unmanaged Device DDT instances with each device. |
| 6 | Activate the **Export** command.<br>**Result**: a progress indicator lets you know how the export is proceeding. |
| 7 | A message in the **output window** tells you that export is complete. |

# Importing a global project

## At a Glance

This function is accessible only from the **File** → **Open** menu of Unity Pro.

The following files can be opened:

- **.ZEF** file (Full Application Exchange Files) to import a global project with global DTMs configuration
- **.XEF** file (Application Exchange Files) to import a global project without global DTMs configuration

Before opening the imported project, Unity Pro proposes to save the current project before closing it.

## Libset Version

When importing a **.ZEF** or **.XEF** file, the latest **Libset** version is always selected.

## Importing protected elements

If the project you want to import contains protected sections, these are imported and their content is coded.

If the project you want to import contains protected DFB types, these are referenced in the .ZEF or .XEF file but not imported.

**NOTE: .ZEF** or **.XEF** files do not contain the configuration of the TSX CPP 110.
If the **.CO** file (**Sycon**) does not exist:

- a warning is displayed in the **Output Window** after the **Open** command,
- an error is displayed in the **Output Window** after the **Analyze** or **Generate** command.

In the configuration screen of the TSX CPP 110 PCMCIA card, select the required **.CO** file to update the CANopen configuration.

## Importing operator screens

Screens are only re-attached to the functional modules to which they were attached before export when you import a complete ZEF, XEF or XFM file.

If you import an operator screen in the structural view (XCR file), the attachment between operator screen and functional module is lost.

If you import an operator screen in the functional view (XCR file), it is attached to the functional module from which the import was launched.

The operator screens are imported with the same structure as in the source project. If families of screens do not exist, they are automatically created.

### Maintain Output Links on Disabled EF

For compatibility of the project setting **Maintain output links on disabled EF (EN=0)** please refer to the respective section *(see page 542)*.

### How to import a global project

To import a global project, perform the following operations:

| Step | Action |
|---|---|
| 1 | Activate the **Open** command from the **File** menu.<br>**Result**: A dialog box appears on the screen. |
| 2 | Choose to save or not save your current project.<br>**Result**:<br>● Your project is saved or not saved, according to your selection.<br>● A dialog box appears on the screen. |
| 3 | Select the file to be imported (.ZEF or .XEF).<br>**Result**: The filename appears in the **File name** field. |
| 4 | Select the **Open import File with Wizard** option, if you require help reassigning elements. |
| 5 | Activate the **Open** command.<br>**Result**:<br>● Unity Pro closes your current project.<br>● A progress indicator lets you know how the import is proceeding. |
| 6 | A message tells you that the import is complete.<br>If any errors occur during import, you are informed of these by a message in the **Output Window**. |

### Conversion Wizard

To convert a legacy project partially or as a whole you can use the **Conversion Wizard**.

The **Conversion Wizard** provides options to remap I/O objects (channels, variables etc.) during conversion and to adapt the hardware configuration of the new project concurrently in Unity Pro.

# Exporting/Importing LL984 Segments/Networks

## Introduction

You can export/import LL984 segments and networks from the **Structural View** of the **Project Browser**.

## Restrictions

In contrast to the **Export** dialog for other language sections, the dialog for LL984 segments/networks does not provide the **With SR** check box, because the subroutine networks called in the exported networks are not exported.

This is done because the called subroutine number can be given as a parameter at runtime, so you do not know in advance which subroutines are really called.

If you want to use the called subroutines in another application, you must export and re-import the whole LL984 subroutine segment.

## Files Generated

Files with the following extensions are generated during export:

- **.X9S* for LL984 segments
- **.X9N* for LL984 networks

The created files also include the definitions of variables and variable types used inside the exported segments and networks.

## Exporting LL984 Segments/Networks

| Step | Action |
|---|---|
| 1 | Select the element to be exported (LL984 segment/network). |
| 2 | Right-click and select **Export** from the context menu or use the **File** → **Export** menu command. |
| 3 | Select the destination directory for the export and enter the file name. |
| 4 | Select / deselect the **With DDT**/**With DFB** options. |
| 5 | Click the **Export** button. |
| 6 | A message in the **Import/Export** tab of the output window informs you when export is completed. |

### Importing LL984 Networks

To import a **.X9N* file (LL984 network) you have to select an LL984 segment node in the **Structural View** of the **Project Browser**

Right-click and select **Import** from the context menu or use the **File** → **Import** menu command.

In case of naming conflicts for the network name or the used variables and variable types an **Import Trouble Report** dialog will be displayed. Please refer to Managing Conflicts *(see page 1728)*.

### Importing LL984 Segments

To import a **.X9S* file (LL984 segment) you have to select **Sections** or **SR Sections** in the **Structural View** of the **Project Browser**.

Right-click and select **Import** from the context menu or use the **File** → **Import** menu command.

In case of naming conflicts an **Import Trouble Report** dialog will be displayed.

Importing a **.X9S* file to the **SR Sections** node can end up in the existence of more than one LL984 segment under **SR Sections**.

During **Analyze** this will be checked and will lead to an analyze error.

In this case move the LL984 subroutine networks needed into one LL984 SR segment and delete the other LL984 SR segments.

Even if the **Ladder Logic (LL984)** option under **Tools** → **Project Settings** → **Program** → **Languages** is not set, **.X9S* files can be imported without error messages and the corresponding segments and networks will be created. But in this case later during **Analyze** an `incorrect language` message will be raised for these segments and networks.

In the context menu of segments the menu entry to import networks is disabled in case the **Ladder Logic (LL984)** option is not set.

# Section 52.3
## Wizard for importing the various elements of Unity Pro

### Object of this sub-chapter

This sub-chapter presents the functions of the wizard for importing the sections and functional modules under Unity Pro.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Using the assistant | 1762 |
| Variables tab of the wizard | 1766 |
| DFB and DDT Tabs of the Wizard | 1768 |
| SR, Sections, Events and Macro steps tabs of the wizard | 1770 |
| Wizard animation tables tab | 1771 |
| Wizard functional module tabs | 1772 |
| Correspondence file | 1774 |

# Using the assistant

## At a Glance

You can select the assistant to import the following elements:

- a global project,
- a section,
- a functional module.

During the import you can use the assistant to facilitate the **reassignment** of the following types of element:

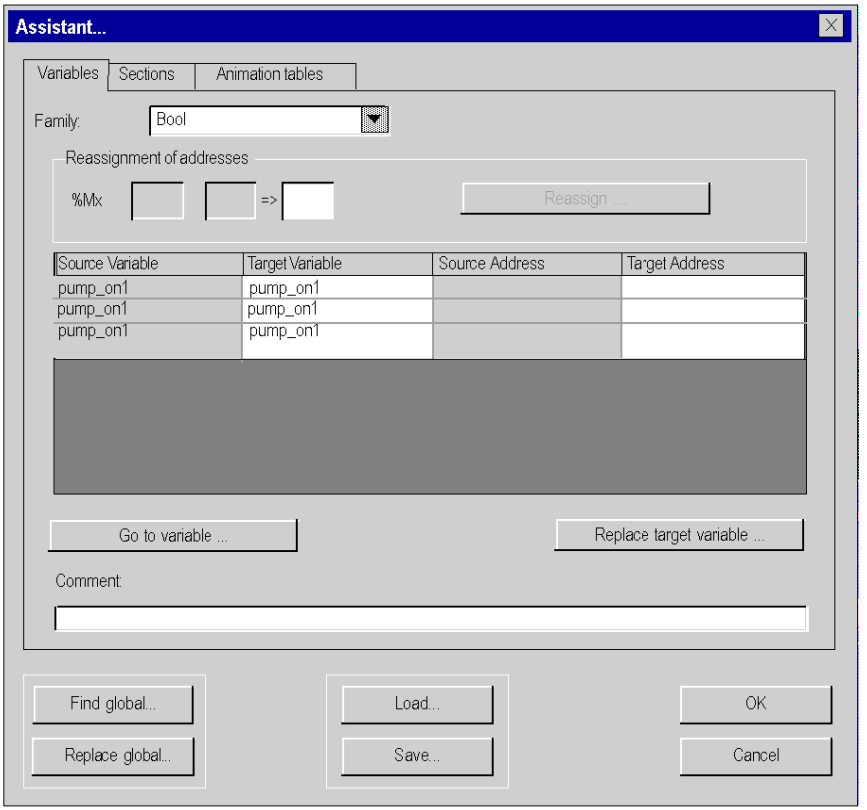
- variables,
- DDTs,
- DFBs,
- sections (SR, macro SFC),
- event processing (timer or Input/Output),
- animation table,
- functional module.

To do this, you must check the **With wizard** box in the import dialog box.

## Description of the wizard

The wizard is accessible via a dialog box. This box contains one or more tabs. Each tab corresponds to an element to be imported.

The illustration that follows is an example of the dialog box when a functional module is imported:



The list of elements that cannot be reassigned (shaded columns) and the contextual functions allowing the reassignments are offered via the following tabs:

| Tab | Description |
|---|---|
| DDT | Provides a list of the DDTs used. For each DDT, the screen gives the name of the DDT before and after reassignment, as well as the type of DDT and the comment. |
| DFB | Provides a list of instances of DFBs used. For each instance, the screen gives the name of the instance before and after reassignment, as well as the DFB type and the comment. |
| Variables | Provides a list of the variables used. For each variable, the screen gives the address and the symbol (before and after reassignment), as well as the comment. |
| Sections | Provides a list of program modules. |
| SR | Provides a list of SRs. |
| Macro steps | Provides a list of the sequential steps called. |
| Events | Provides a list of Timer or System event processing elements called. |
| Animation tables | Provides a list of animation tables used. |
| Functional modules | Provides a list of elements (variables, sections, animation tables) that are in the functional module to be imported. |

## Common commands

All the tabs of the wizard contain the same commands:

- Find global and Replace global *(see page 1767)*,
- Load and **Save** *(see page 1774)*.

## New addresses reassignment using the wizard

The procedure below explains how to reassign a range of addresses using the import assistant wizard, note that only symbolic variables can be modified using the assistant wizard:

| Step | Action |
|---|---|
| 1 | Select the **Variable** tab from the wizard window. |
| 2 | Select the first source address from the range of addresses to re-allocate. |
| 3 | Right-click on the first source address. |
| 4 | Select **Set Low Limit**.<br>**Result:** In the **Address remapping** section, the low limit is set. |
| 5 | Select the last source address from the range of addresses to re-allocate. |
| 6 | Right click on the last source address. |
| 7 | Select **Set High Limit**.<br>**Result:** In the **Address remapping** section, the high limit is set. |

| Step | Action |
|---|---|
| 8 | In the blank field from the **Address remapping** section:<br>● Enter the first new remapping address. |
| 9 | Click on **Remapping...**.<br>**Result:**<br>● The original range of addresses is updated.<br>● The **Target Address** column is updated with the new range of addresses. |

# Variables tab of the wizard

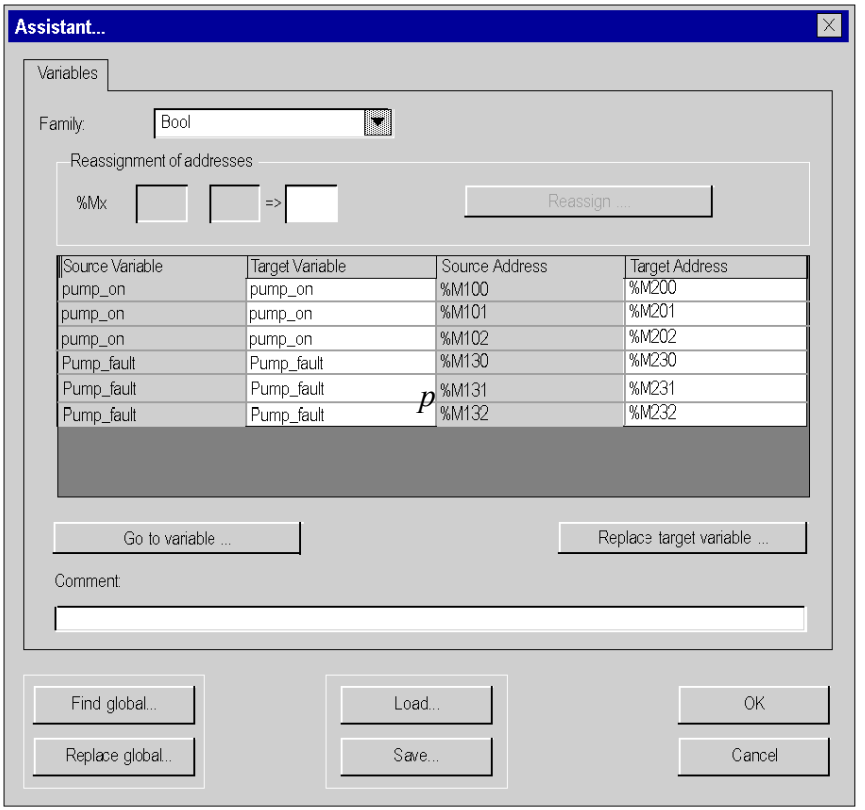
## At a Glance

This tab allows you to edit the name of the symbols or to reassign an object or a group of objects to be imported.

If you want to import input/output variables, this editing allows you to reassign them in order to adapt to the new current configuration.

## Reassignment of variables

The **Reassign** command allows you to reassign a group of variables. For example, you can reassign the objects %M100...%M132 into %M200...%M232:

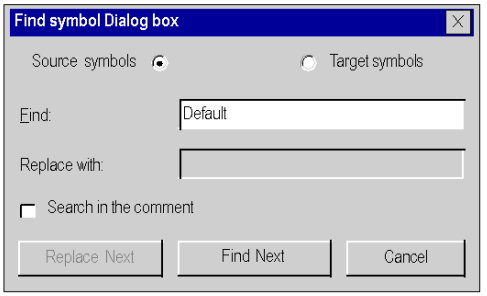



## Sort by type

In the **Family** field you can select the type of variables to display in the window.

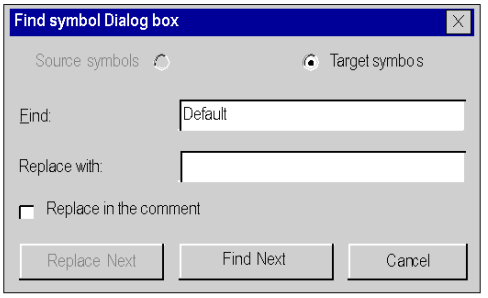
## Modifying a symbol

The **Go to variable** command allows you to find a symbol or a character string in the symbols (source or destination):




If you check the **Search in the comment** box, the variable search is also launched in the variable comments.

The **Replace target variable** command allows you to modify a symbol or replace a character string in the symbols to be imported:



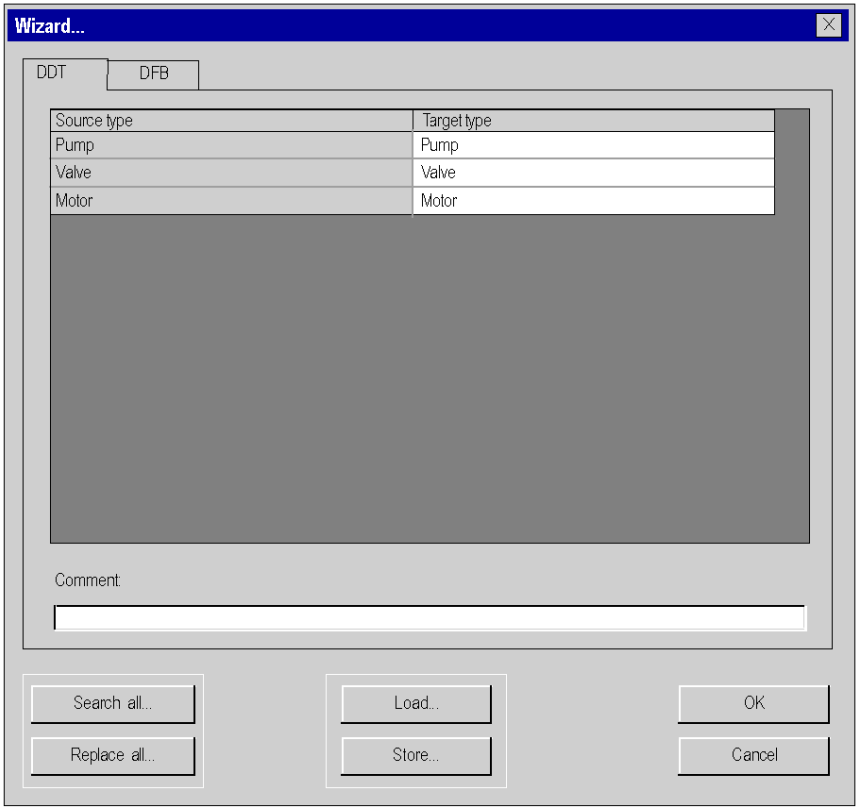

If you check the **Replace in the comment** box, the variable will also be replaced in the variable comments.

**NOTE:** The **Go to variable** and **Replace target symbols** commands are available in all the other tabs during import with the wizard. They are available with the command buttons **Find global** and **Replace global**.

# DFB and DDT Tabs of the Wizard

## At a Glance

These Two Tabs Present the DFB or DDT Types to Be Imported.




## Modifying an Element Name

To modify the name of an element, you must perform the following operations:

| Step | Action |
|---|---|
| 1 | Double click on the element to be modified. |
| 2 | Input the new name of the element. |
| 3 | Confirm your input by pressing the **Enter** button. The **Escape** button cancels the input. |

### Elements You May Modify

The various elements that you are able to modify are the following:

- DFB:
  - **Target type** enables you to modify the name (**Source type**) of a DFB type. This name may contain a maximum of 32 alphanumeric characters (‘A’...‘Z’, ‘0’...‘9’ and underscore character); the first character must be a letter.
  - The new name of a type must be unique.
  - The DFB type may not be modified.
  - **Comment** enables you to modify comment associated with a selected DFB type. This comment, written on a single line, may contain a maximum of 1024 characters.
- DDT:
  - **Target type** enables you to modify the name (**Source type**) of a DDT type. This name may contain a maximum of 32 alphanumeric characters (‘A’...‘Z’, ‘0’...‘9’ and underscore character); the first character must be a letter.
  - The new type names must be unique.
  - **Comment** enables you to modify comment associated with a selected DDT type. This comment, written on a single line, may contain a maximum of 1024 characters.

# SR, Sections, Events and Macro steps tabs of the wizard

## At a Glance

These four tabs present the sections, SR program modules, event processing elements and macro steps to be imported.

## Modifying the name of an element

To modify the name of an element, you must perform the following operations:

| Step | Action |
|---|---|
| 1 | Double click the element to be modified. |
| 2 | Enter the new name for the element. You can only modify the **name** of a program module or of a macro step and the **number** of an event processing element. |
| 3 | Confirm your input by pressing **Enter**. Press **Esc** to cancel the input. |

## Elements that you can modify

The various elements that you can modify are as follows:

- SR program sections and module
  - **Target Address** allows you to modify the name of an SR program section or module. This name comprises a maximum of 32 alphanumeric characters (‘A’...’Z’, ‘0’...’9’ and underscore character); the first character is a letter.
  - **Comment** allows you to modify the comment associated with the SR program section or module. This comment written on a single line may contain a maximum of 512 characters.
  - The new SR program section or module must be unique.
- Event processing
  - **Target Address** allows you to modify the event processing number.
  - **Comment** allows you to modify the comment associated with the event processing element. This comment written on a single line may contain a maximum of 512 characters.
  - The new event processing must be unique.
- Macro step
  - **Target Address of the Macro** allows you to modify the name of a macro step.
  - **Comment** allows you to modify the comment associated with the macro step. This comment written on a single line may contain a maximum of 512 characters.
  - The new macro step must be unique.

# Wizard animation tables tab

## At a Glance

This tab presents the animation tables to be imported.

## Modifying the name of an element

To modify the name of an element, you must perform the following operations:

| Step | Action |
|---|---|
| 1 | Double click the element to be modified. |
| 2 | Enter the new name for the element. You can only modify the **name** of an animation table. |
| 3 | Confirm your input by pressing **Enter**. Press **Esc** to cancel the input. |

## Elements that you can modify

The various elements that you can modify are as follows:

- Animation table
  - **Target address** allows you to modify the name of the animation table. This name comprises a maximum of 32 alphanumeric characters (‘A’...’Z’, ‘0’...’9’ and underscore character); the first character is a letter.
  - **Comment** is used to modify the comment associated with the animation table.
  - The new animation table must be unique.

# Wizard functional module tabs

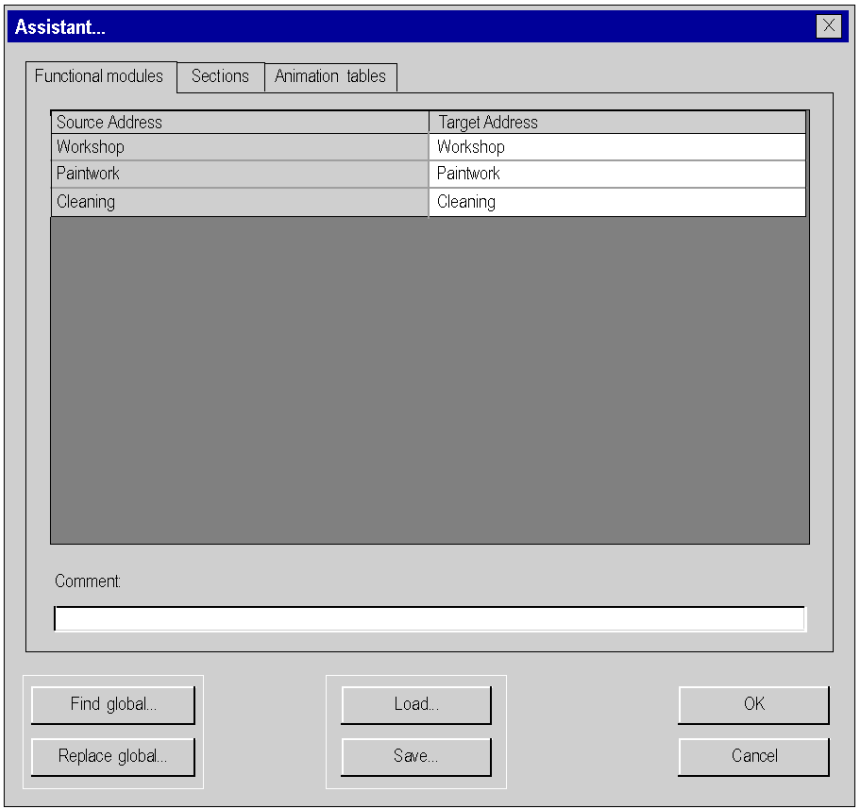
## At a Glance

When importing a functional module with the wizard, the dialog box appears with the following tabs:

- Functional modules,
- Variables *(see page 1766)*,
- Sections *(see page 1770)*.
- Animation tables *(see page 1771)*,
- etc.

The tab of the element to be imported appears only if this element exists in the functional module.

The following illustration presents the dialog box when a functional module is imported:



### Elements you can modify

The various elements you can modify are as follows:

- Functional module:
  - **Target Address** allows you to modify the target name of the functional module. This name comprises a maximum of 32 alphanumeric characters (‘A’...’Z’, ‘0’...’9’ and underscore character); the first character is a letter.
  - The new functional module name must be unique.
  - **Comment** is used to modify the comment associated with the selected functional module. This comment written on a single line may contain a maximum of 1024 characters.

# Correspondence file

## At a Glance

This file in text format contains all the information concerning the modifications or reassignments made in the various tabs of the wizard: Variables, DFB, DDT, Sections, Macro steps, etc.

| ⚠ CAUTION |
| --- |
| **LOSS OF DATA**<br>Save the Correspondence file to disk before launching the import command.<br>**Failure to follow these instructions can result in injury or equipment damage.** |

## Saving/Restoring modifications

The **Load** and **Save** commands allow you to:

| Command | Description |
| --- | --- |
| Load | Allows you to automatically restore the reassignments previously saved to a file. |
| Save | Allows you to save the modifications made, via the wizard, to a correspondence file.<br>The following dialog box allows you to enter the name of the file:<br>  |

# Part X
## Conversion Wizard

# Chapter 53
# Conversion Wizard

## Overview

This chapter describes the conversion wizard.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 53.1 | General Description of the Conversion Wizard | 1778 |
| 53.2 | Structure Tab of the Conversion Wizard | 1787 |
| 53.3 | I/O Remapping Tab of the Conversion Wizard | 1792 |
| 53.4 | CPU Configuration Tab of the Conversion Wizard | 1813 |

# Section 53.1
# General Description of the Conversion Wizard

## Overview

This section provides a general description of the conversion wizard and how to access the wizard.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction | 1779 |
| Access to Conversion Wizard | 1781 |
| General Representation | 1784 |
| Conversion and Result | 1786 |

# Introduction

## Overview

The conversion wizard is an integrated part of Unity Pro.

The conversion wizard is used to:

- convert applications, exported out of legacy applications (PL7 and Concept) to Unity Pro
- convert legacy applications partially or as a whole
- remap I/O objects (channels, variables etc.) during conversion by means of the wizard
- adapt the hardware configuration of the new application concurrently in Unity Pro
- modify the amount of used memory in the CPU

The conversion wizard is available if you have chosen to install a converter (e.g. PL7 Application Converter) during the setup of Unity Pro.

## Application Conversion as a Whole

To convert an application as a whole, while keeping the same PLC family and selection of application parts or remapping of I/O objects is not needed, you have to use the specific converters directly via the Unity Pro menu **File** → **Open**.

Additional documentation is provided for the specific converters (*Concept Application Converter* and *PL7 Application Converter*).

## Partial Application Conversion

To convert an application partially and/or the PLC family must be changed or remapping of I/O objects is needed, use the conversion wizard via the Unity Pro menu **Tools** → **Convert Partially**.

## General Procedure

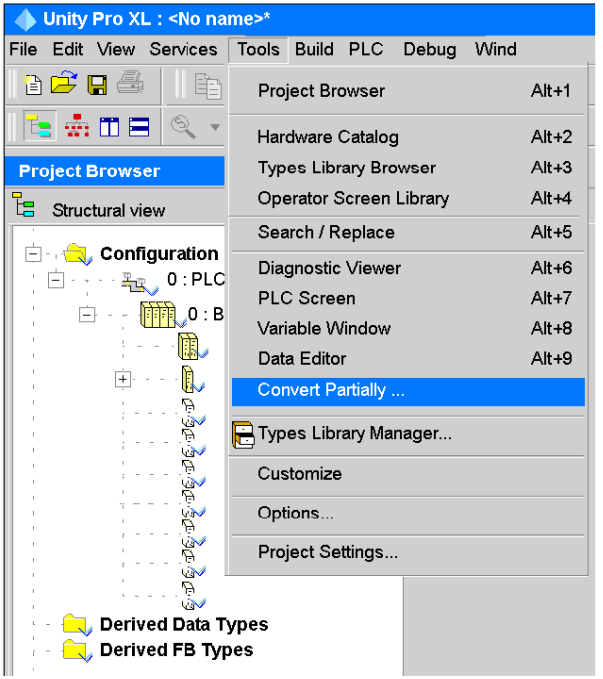
General procedure to convert a legacy application to Unit Pro

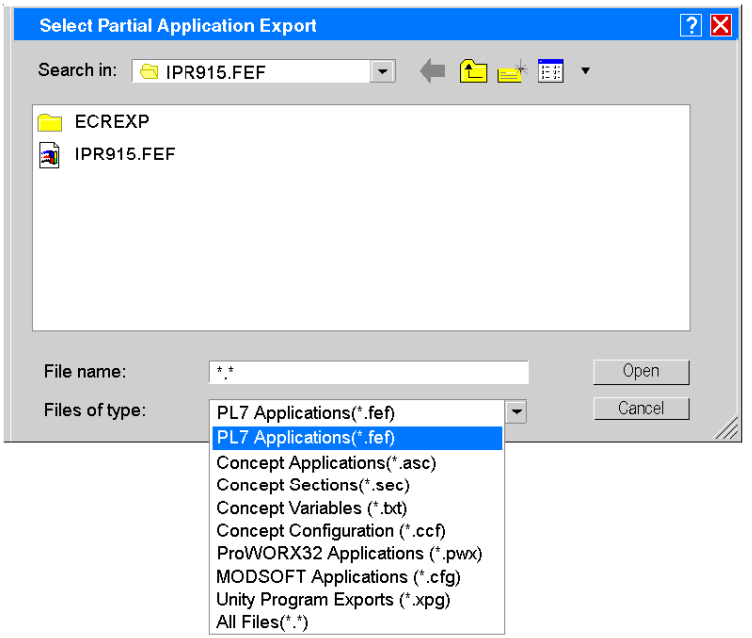
| Step | Action |
|---|---|
| 1 | Export your application out of your legacy programming system (e.g. as an FEF file out of PL7 or an ASC file out of Concept). |
| 2 | Create a new application in Unity Pro selecting a CPU with enough memory and the I/O access capabilities needed (e.g. Modicon M340).<br>Optionally you can configure the I/O modules expected to be needed but you can modify the hardware configuration even later (see step 6). |
| 3 | Launch the conversion wizard in Unity Pro via **Tools** → **Convert Partially**.<br>**Result:** The conversion wizard asks you to select the exported legacy source file. |
| 4 | Select the exported legacy source file.<br>**Result:** The converter analyzes the source file and displays the result in the 3 tabs of the conversion wizard. |
| 5 | Select the parts of the application (or the complete application) to be converted in the **Structure** tab. |

| Step | Action |
|---|---|
| 6 | Remap the I/O objects for getting them compliant with the new hardware configuration.<br>Concurrently you can modify the hardware configuration of the new application in Unity Pro.<br>**Note:** To save a backup file of your intermediate I/O mapping you can use the **Save** button. With **Load** you can reload your latest saved intermediate I/O mapping. |
| 7 | After finishing all your selections and manual modifications click **OK**.<br>**Result:** The converter applies the defined remapping to the selected parts of the source file and imports the results into the opened Unity Pro Application (e.g. an application with an Modicon M340 PLC). |
| 8 | Continue working on the opened application, save it or export it as a ZEF or XEF file *(see page 1755)*. |

# Access to Conversion Wizard

## Accessing the Wizard

| Step | Action |
|---|---|
| 1 | Create a new application in Unity Pro selecting a CPU with enough memory and the I/O access capabilities needed (e.g. Modicon M340). |
| 2 | Launch the conversion wizard in Unity Pro via **Tools** → **Convert Partially**.   |

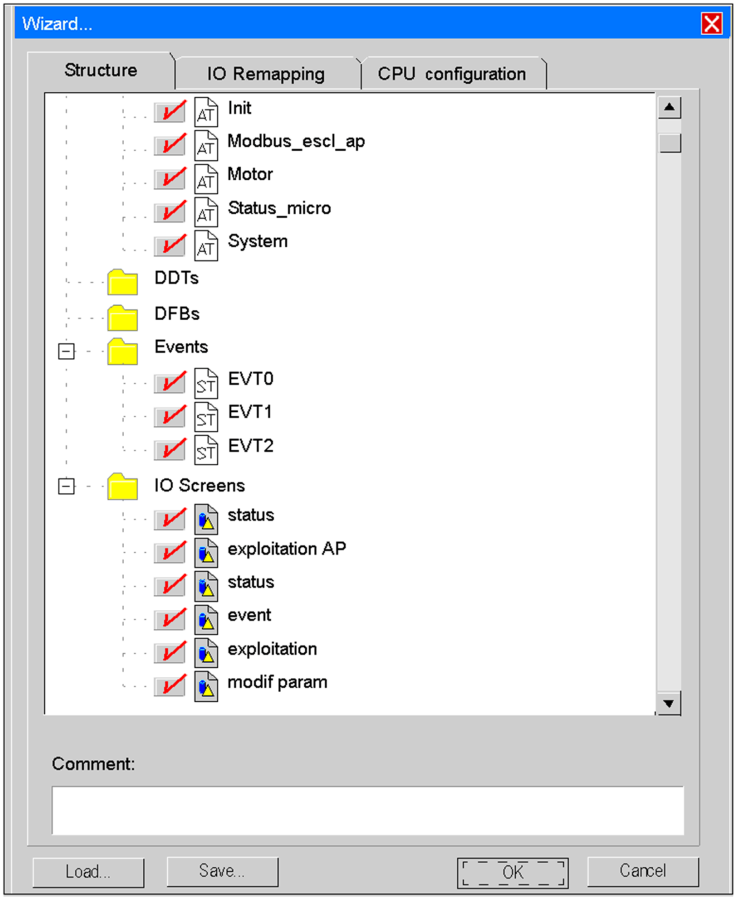
| Step | Action |
|---|---|
| 3 | A file selection dialog is displayed, offering a choice of exported legacy application files (e.g FEF files for PL7, ASC files for Concept etc.).<br>  |
| 4 | Select a file and confirm with **Open**.<br>**Result:** Depending on the kind of file selected the respective converter (PL7, Concept, LL984) becomes active and analyzes the content of the selected file. The result of the analysis is displayed in the 3 tabs of the conversion wizard. |

## Exported Files

| Software | Extension | Exported |
|---|---|---|
| PL7 | *.fef | complete application |
| Concept | *.asc | complete application |
| Concept | *.sec | user-defined subset of sections |
| Concept | *.txt | user-defined subset of variables |
| Concept | *.ccf | configuration export to extract communication definitions |
| ProWORX32 | *.pwx | complete application |
| MODSOFT | *.cfg | complete application |
| Unity Pro | *.xpg | tasks (without configuration) |

# General Representation

## Overview



## Elements of the Wizard

| Element | Description |
|---|---|
| tabs | The conversion wizard shows 3 tabs:<br>• **Structure**<br>In this tab you can select the parts of the application to be converted.<br>Please refer to *Structure Tab of the Conversion Wizard, page 1787*.<br>• **IO Remapping**<br>In this tab you can remap I/O objects before conversion.<br>Please refer to *I/O Remapping Tab of the Conversion Wizard, page 1792*.<br>• **CPU configuration**<br>In this tab you can modify the amount of used memory in the CPU.<br>Please refer to *CPU Configuration Tab of the Conversion Wizard, page 1813*. |
| **Save...** | Saves all your input made in the different tabs, as selecting the parts to be converted or remapping of I/O objects, to a backup file. Each **Save...** will overwrite your last backup file. |
| **Load...** | Restores all input, made in the different tabs, from a backup file.<br>There is only 1 level of backup file created with your last **Save...**. |
| **OK** | Starts conversion taking into account all the input made in the different tabs of the wizard. |
| **Cancel** | Cancels conversion and closes the wizard. |

The content of the tabs may differ depending on the source code that is converted.

# Conversion and Result

## Starting Conversion

Click the **OK** button to start converting the already prepared application.

## XPG File

An XPG file containing all user corrections is saved during conversion.

- In case of PL7 conversion an XPG file is saved to a user-related directory:
  *My Documents\PL7 Converter\<app name>\<app name.xpg>*
  This directory is different from the directory where the input file has been taken from. This allows to read also from network directories with write-protection.
- In case of Concept and LL984 conversion an XPG file is saved to the same directory as where the source file is located.

**NOTE:** XPG format is used as output file format because it contains programs only, but no configuration.

## Automatic Import

The import of the converted application into the opened application is automatically launched to free you from an otherwise additional operation step.

## Report File

A report of conversion that gives the list of new mapping, warnings and what is not converted, accompanies the XPG output file in the same user-related directory.

## Application Build

To be able to build the application in Unity Pro correct the I/O mapping completely and remove all other errors displayed during analysis of the application.

# Section 53.2
## Structure Tab of the Conversion Wizard

### Overview

This section provides a description of the conversion wizard´s **Structure** tab and how to select the parts of the application to be converted.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction | 1788 |
| Selecting Application Parts to be Converted | 1790 |

# Introduction

## Overview

The **Structure** tab of the conversion wizard shows the analyzed legacy application in a browser view.

In this browser you can select all parts of an application that should be converted.

## Selectable Parts

The following parts of an application can be selected for conversion:

- program tasks and sections of programs in all programming languages, subset list of sections
- subset list of subroutines
- subset list of animation tables
- subset list of DDTs and DFBs
- subset list of events
- subset list of I/O screen families and I/O screens (operator screens)
- variables

Depending on the legacy project, not all parts may exist.

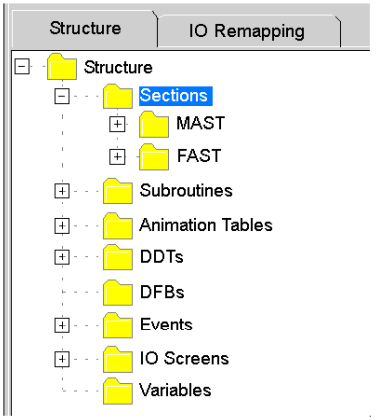
**NOTE:** By default all parts of an application are selected for conversion.
**I/O Screens** (operator screens) can not be selected.

## Variables

If you select the **Variables** part for conversion, all the variables that are declared data (having a symbol or a comment) are taken into account during conversion.

Otherwise only data, called inside the selected sections, subroutines and events, are converted.

## Browser View




The extension sign (+) in front of a directory indicates that there are selectable subdirectories.

# Selecting Application Parts to be Converted

## Overview

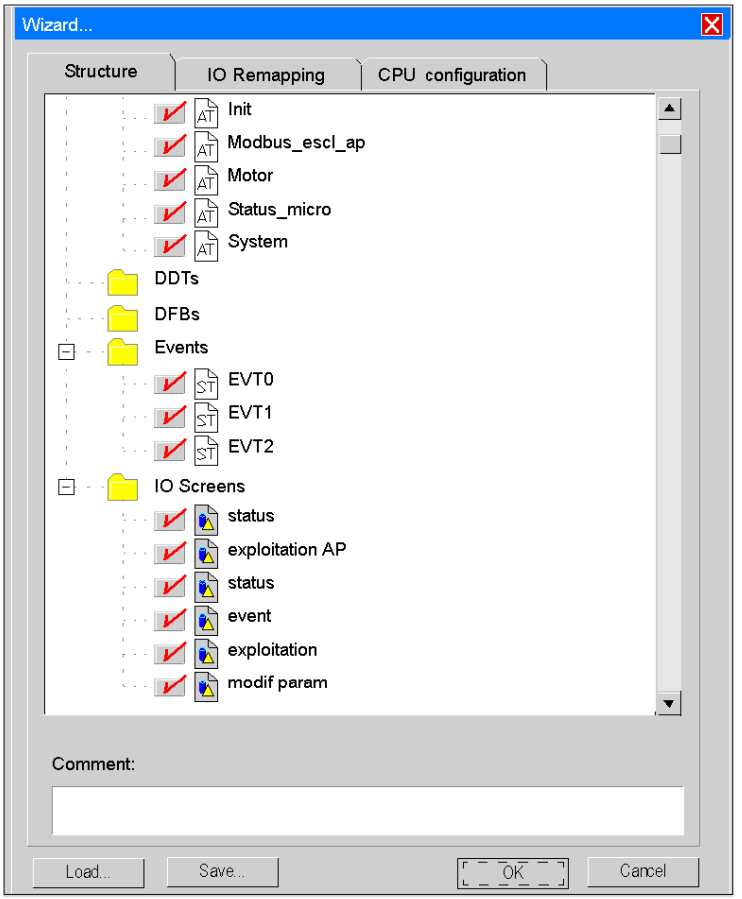
The **Structure** tab of the conversion wizard shows the analyzed legacy application in a browser view.

In the browser view you can select the parts of an application to be converted.

You can either select single parts or all parts of a directory at once.

You can not select a directory itself.

## Structure Tab



## Elements of the Tab

| Element | Description |
|---|---|
| checkbox | A checkbox with a red checkmark indicates that this part should be converted.<br>An empty checkbox indicates that this part should not be converted.<br>Click the checkbox or the icon beside to toggle from checked to unchecked and vice versa. |
| **Comment** | Displays the comment of a selected part in the browser view (read only). |

## Context Menu

Right click on an application part in the browser view to open a context menu.

This context menu provides the **Select All** and the **Select None** items.

| Element | Description |
|---|---|
| **Select All** | Click on any part of a directory and then click **Select All** to select all parts of the respective directory. |
| **Select None** | Click on any part of a directory and then click **Select None** to select all parts of the respective directory. |

# Section 53.3
## I/O Remapping Tab of the Conversion Wizard

### Overview

This section provides a description of the conversion wizard´s **I/O Remapping** tab and how to remap I/O objects before conversion.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction | 1793 |
| Remapping Table | 1796 |
| Remapping Channels, Groups of Channels or Specific I/O Types | 1799 |
| Clustering and Remapping of Memory Variables | 1801 |
| Memory Area Tabs of the Conversion Wizard | 1803 |
| Concurrent Editing | 1805 |
| Exclude, Communication -1, Type Restriction | 1808 |

# Introduction

## Overview

The **I/O Remapping** tab of the conversion wizard shows the I/O objects of an analyzed legacy application in a table view.

By means of this table view you can remap I/O objects before conversion.

The **I/O Remapping** tab provides additional features to ease special remapping operations (**Communication -1**, **Exclude**, **Type Restricted**).

The **I/O Remapping** tab is applicable for all applications coming from PL7, Concept or ProWORX32.

## Syntax of I/O Objects

For the detailed syntax of I/O objects and their correspondences in Unity Pro please refer to the specific documentation to be found under:

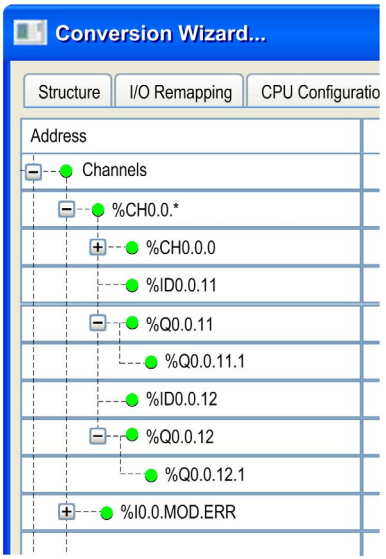
- *Concept Converter -> Requirements and conversion -> Requirements -> System*
- *PL7 Converter -> Correspondence between PL7 and Unity Pro -> Correspondences between common language elements -> Correspondences between PL7 and Unity Pro: language objects*

## Address Column

Directly addressed I/O variables (topological addresses) are represented in the **Address** column of the conversion wizard as shown in the following example.

Please also refer to *PL7 Converter -> PL7 application conversion procedure -> Conversion of a PL7 application -> Topological Addresses of PL7 Micro in the Conversion Wizard*.

Embedded counter channels of a PL7 TSX Micro CPU




For detailed address description please refer to *Languages Reference -> Data Description -> Data Instances -> Direct Addressing Data Instances*.

## Corrected Column

The addresses in the **I/O Remapping** tab are formally adapted to Unity Pro syntax, but have the same numerical values as before in PL7.

For details please refer to *PL7 converter -> Correspondence between PL7 and Unity Pro -> Correspondences between common language elements -> Correspondences between PL7 and Unity Pro: language objects -> Memory objects (variables and constants)*.

In the **Corrected** column only direct addresses following the Unity Pro syntax may appear.

If modules do not exist in Unity Pro, existing I/O objects have to be mapped manually to other regular modules of the targeted PLC.

## I/O Object Remapping

**I/O Remapping** is applicable to all selectable source formats and PLC families coming from PL7, Concept or ProWORX32.

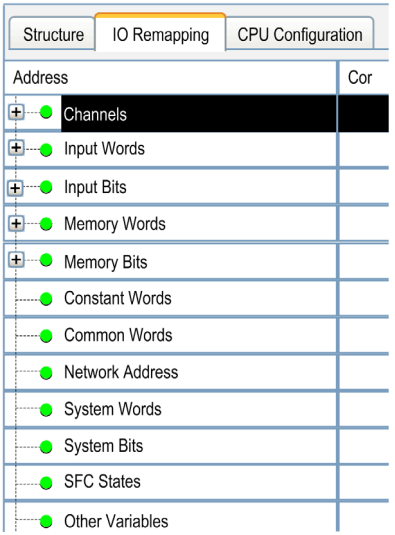
## Variable Families

Because applications can have very many variables, the variables are structured into families.

A variable family is characterized by direct addresses of the same semantically meaning, e.g. Word addresses or addresses of boolean variables.

## Table View

The names of all variable families are listed in the table view.




If there is no extension sign (+) in front of a variable family, no object of this family is present in the application.

# Remapping Table

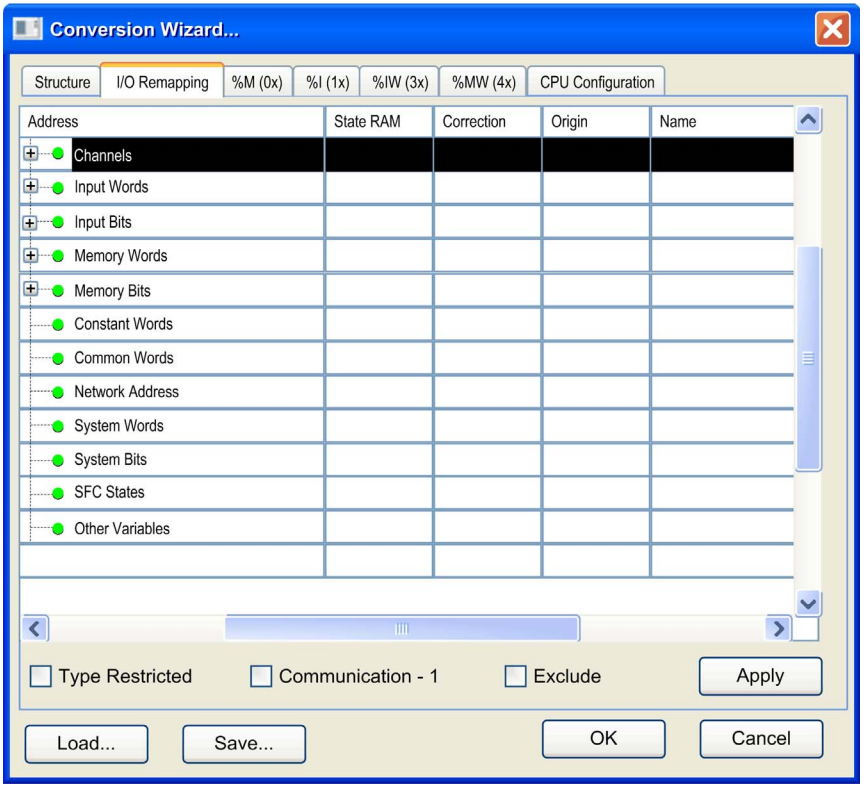
## Overview

The remapping table shows all variables with direct addresses used in the application whether they are declared or not.

Even if directly addressed variables appear only in programs, they are detected and displayed in this table.

## Representation

**I/O Remapping** tab



## Elements of the Remapping Table

To see all columns of the table scroll horizontally.

| Element | Description |
|---|---|
| **Address** | Topological or direct variable address<br>For detailed information please refer to paragraph below. |
| **State RAM** | This column is only available for Concept, Modsoft, ProWORX32 or ProWORX Nxt applications with State RAM assignments.<br>If the **State RAM** column is visible, the **Correction** column only accepts State RAM addresses. |
| **Correction** | In this column you can enter corrections for the values of the **Address** column. Use the same format as in **Address** column.<br>Example:<br>**Address:** `%I\1.1\1.4.1` -**Correction:** `%I\1.1\1.3.1` |
| **Origin** | Source module of the imported legacy application |
| **Section** | A comma-separated list of section names in which the variable is used or the description of the hardware module, if available in the application file. |
| **Name** | Symbolic name |
| **Type** | Variable type |
| **Comment** | Variable comment |
| **Apply** | Starts the defined remapping for the currently selected variables. |
| **Communication -1** | Please refer toExclude, Communication -1, Type Restriction *(see page 1808)*. |
| **Exclude** | |
| **Type Restricted** | |

## Address

For data that have hierarchical elements, like channels for hardware modules in TSX Premium and TSX Micro, there is an extension sign (+) to expand the list. For example, an overview of the channels of the hardware modules is displayed.

The **Address** column shows the hierarchy of the topological addresses in tree form. The rows are sorted according to the module addresses (e.g. `%CH0.1*`), which are shown as:

`%CH{\<Bus alias>.<Connection point>\}<Rack>.<Module>.*`

The module addresses are derived from the hardware modules present in the configuration part of the application file.

The addresses are composed with a wildcard character "`.*`". The wildcard character stands for all channels of the hardware module (e.g. `%CH0.1*`).

In the next level (e.g. %CH0.6.1.*), that can be expanded by clicking the extension sign (+) in front of a module row, the channels of the module are listed.

This is done by additionally showing the channel number, followed by a wildcard character ".*". The wildcard character stands here for the various data of the channel, which can have more than 1 level.

Below the **Channels** item, there are the items for **Input** and **Memory** that are both subdivided into **Words** and **Bits**. These 4 groups directly represent the State RAM of the PLC. They only show those variables that are used in the entire program. Unused I/O variables are not displayed here.

The 2 **Memory** groups contain clusters of memory items with consecutive addresses. In these groups it is allowed to rearrange consecutive areas. This is not allowed in the 2 **Input** groups.

### Correction

You can define new start addresses for the memory items in the **Memory Words** and **Memory Bits** groups. The new addresses are displayed in the **Correction** column of the **I/O Remapping** tab. For further information on the remapping process, refer to the chapter Clustering and Remapping of Memory Variables *(see page 1801)*.

### Section

The **Section** column shows a comma-separated list of section names in which the variable is used or the description of the hardware module, if available in the application file.

# Remapping Channels, Groups of Channels or Specific I/O Types

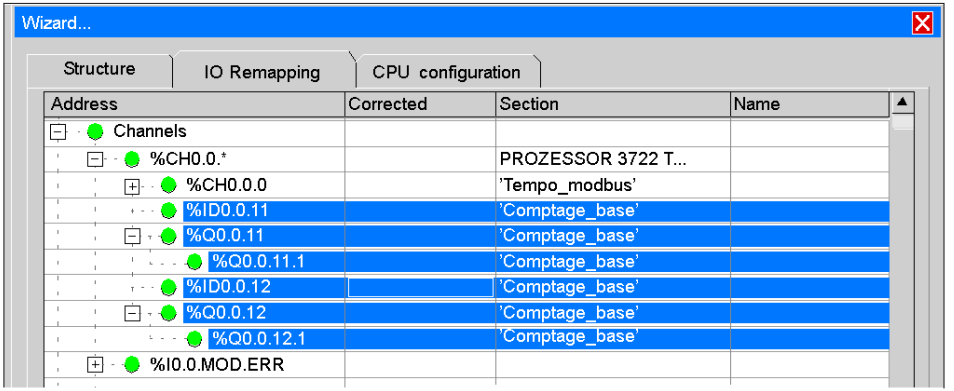
## Overview

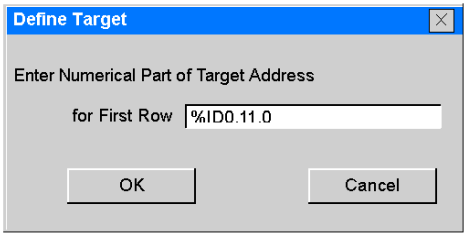
There are several kinds of objects that are managed in hardware modules, for example %IW, %QW, %MW, %I, %Q, etc.

By means of the conversion wizard you are able to

- move all objects from one slot to another slot.
  This is equivalent to search any kind of objects with e.g. addresses `\4.0\2.*` and replace in all kind of objects with `\5.2\3.*`.
- move a subset of channels from one location to another one.

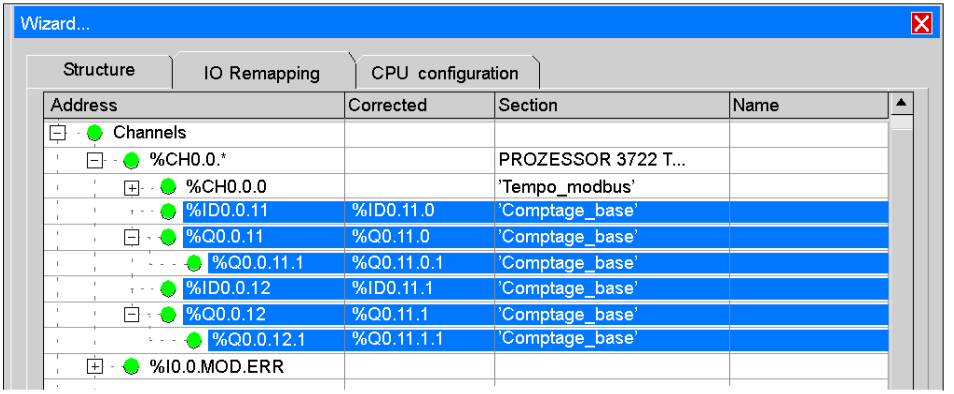
## Procedure

| Step | Action |
|---|---|
| 1 | Select the rows of the data to be remapped either at the channels level or in levels below.<br>  |
| 2 | Right click anywhere in the selected area to open the context menu.<br>  |

| Step | Action |
|---|---|
| 3 | Select **Define Target** from the context menu.<br>The **Define Target** dialog opens.<br>  |
| 4 | Enter the address that should be assigned to the first selected row in the remapping table. |
| 5 | Click **OK** to confirm.<br>**Result:**<br>The selected rows are remapped incrementally starting from the new address entered for the first selected row. Even addresses in levels below are remapped.<br>The new addresses are displayed in the **Corrected** column. See example below. |

## Example

Embedded counter channels of a PL7 TSX Micro CPU moved to a counting module of a Modicon M340




## Validity Check

There is no validity check of the new mapping with the opened application during editing.

The check of consistency is done after converting the application, during the analysis of the application in Unity Pro.

# Clustering and Remapping of Memory Variables

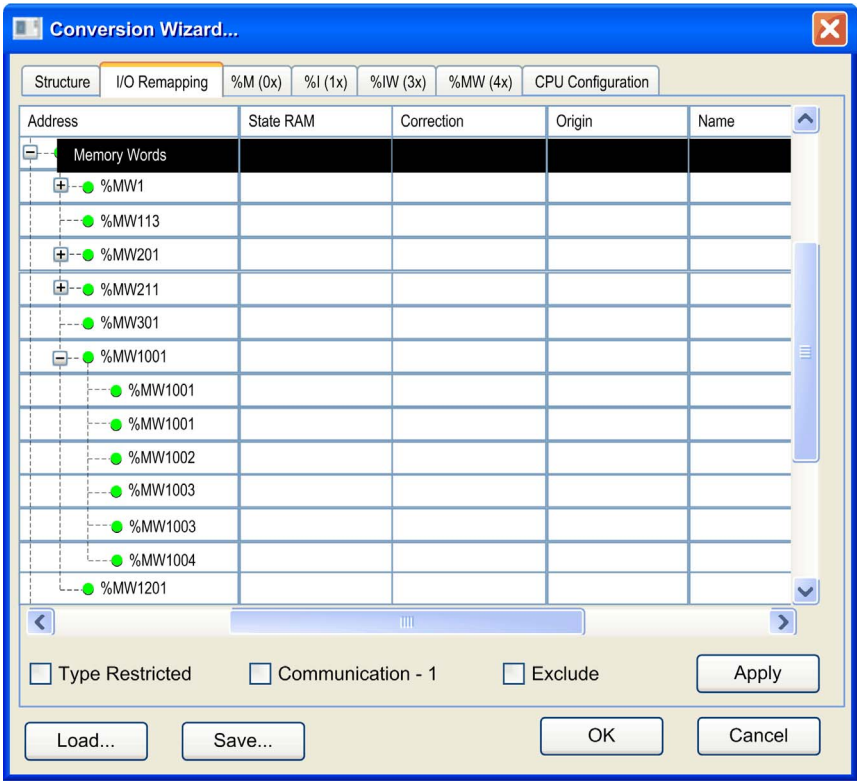
## Overview

Not only topological addressed I/O objects, but also directly addressed memory is structured similarly into clusters.

A cluster is a sequence of memory cells, in which immediate neighbor cells with an address difference of 1 are collected.

## Representation

The graphic below shows a cluster example with `%MW1001`.



## Remapping

Remapping can be applied to channels and such clusters.

Subsequent sets of clusters, subsequent sets of elements on lower levels, single clusters or single elements can be selected and remapped by invoking the command from the context menu. The new address will be displayed in the **Correction** column.

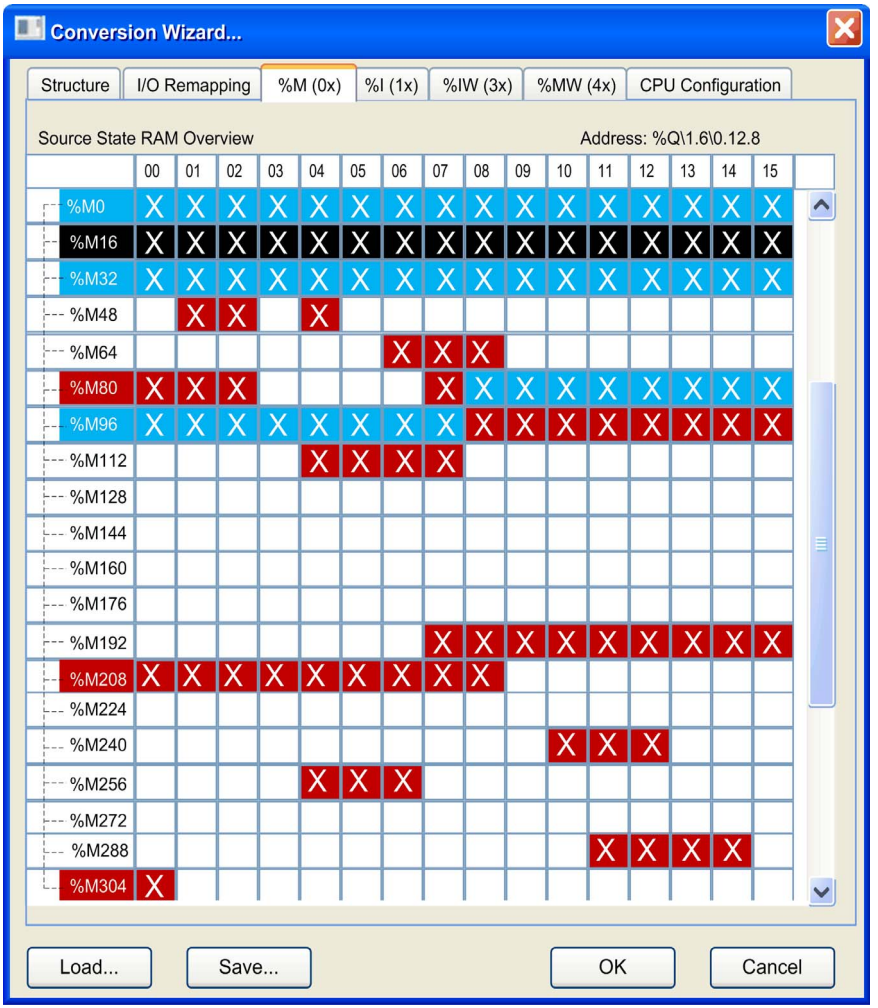
Alignment of variables with type `DINT`, `UDINT` and `REAL` is automatically performed, if a Modicon M340 configuration has been initially defined.

# Memory Area Tabs of the Conversion Wizard

## Overview

The memory area tabs (**%M (0x) to %MW (4x)**) are only available for Concept, Modsoft, ProWORX32 or ProWORX Nxt applications with State RAM assignments. They show the variable allocation in the State RAM of the source application.

## Memory Area Tabs (%M (0x) to %MW (4x))



The memory area tabs (**%M (0x) to %MW (4x)**) indicate the variable allocation of the State RAM in the source application.

The elements of the dialog are color-coded as follows:

| Element | Color |
|---|---|
| module allocated variables | blue |
| language allocated variables | red |

In this dialog, module allocated variables are dominant. This means that only module allocation is indicated in the dialog, even if the variable is language accessed.

Select a cell in the table to indicate the topological address of the variable in the upper right corner of the dialog.

To display different memory areas, select the respective memory area tab (**%M (0x), %I (1x), %IW (3x), %MW (4x)**).

# Concurrent Editing

## Overview

The **Conversion Wizard** is a modeless window, i.e. you can work either in the **Conversion Wizard** or for example in the Unity Pro hardware configuration window.

Therefore you can adapt the remapping and modify the hardware configuration concurrently.

- You can adapt the default mapping done by the converter according to the real hardware.
- You can concurrently modify the configuration of the new Unity Pro hardware.
- You can adapt the hardware configuration to the legacy application, while remapping I/O objects according to the modularity of the modules that may be different to legacy hardware.

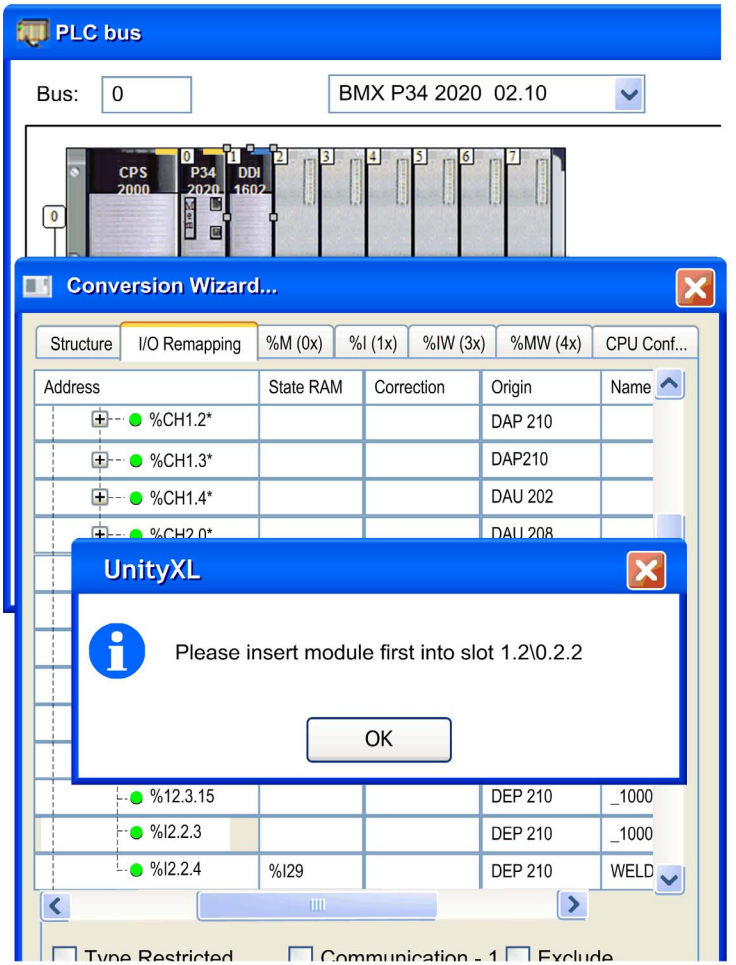
## Initial Situation

A new Modicon M340 configuration appears in the Unity Pro configuration editor.

The application of which parts have to be added is opened in the **Conversion Wizard**.

The I/O channels used in the application appear in the **I/O Remapping** tab.

## Representation



## Interactive I/O Remapping

| Step | Action |
|---|---|
| 1 | Select a channel group representing a module, e.g. CH1.1.* |
| 2 | Enter the corresponding target module into the configuration editor. |
| 3 | Open the target address dialog for the chosen channel group. |
| 4 | Enter the new channel assignment. |

**NOTE:** It is required to enter a module to which a correction order refers into configuration before giving the referring correction command.

The **Conversion Wizard** checks the Unity Pro configuration for the presence of the module - if it is not there, the correction is not executed, leaving the target field blank.

If the module is present in the configuration, the **Conversion Wizard** reads out its State RAM assignments done by the Unity Pro configuration and replaces the assignments in the State RAM column by the new addresses.

Internally it keeps the old addresses and forms correction commands to the Unity Pro-internal correction processor, which replaces all occurrences of the old addresses with the new addresses, when acknowledgement is given with the **OK** button.

Corrections for all modules can be collected before acknowledging, giving the ability to correct also previous corrections.

## No Automatic Notification

Even though you can adapt the remapping and modify the hardware configuration concurrently, there is no automatic notification between hardware configuration and **Conversion Wizard** about changes on either side.

**NOTE:** Following the conversion of the project, some adjustments may be needed depending on the differences of behavior between the original hardware module and the selected module in the Unity Pro platform.

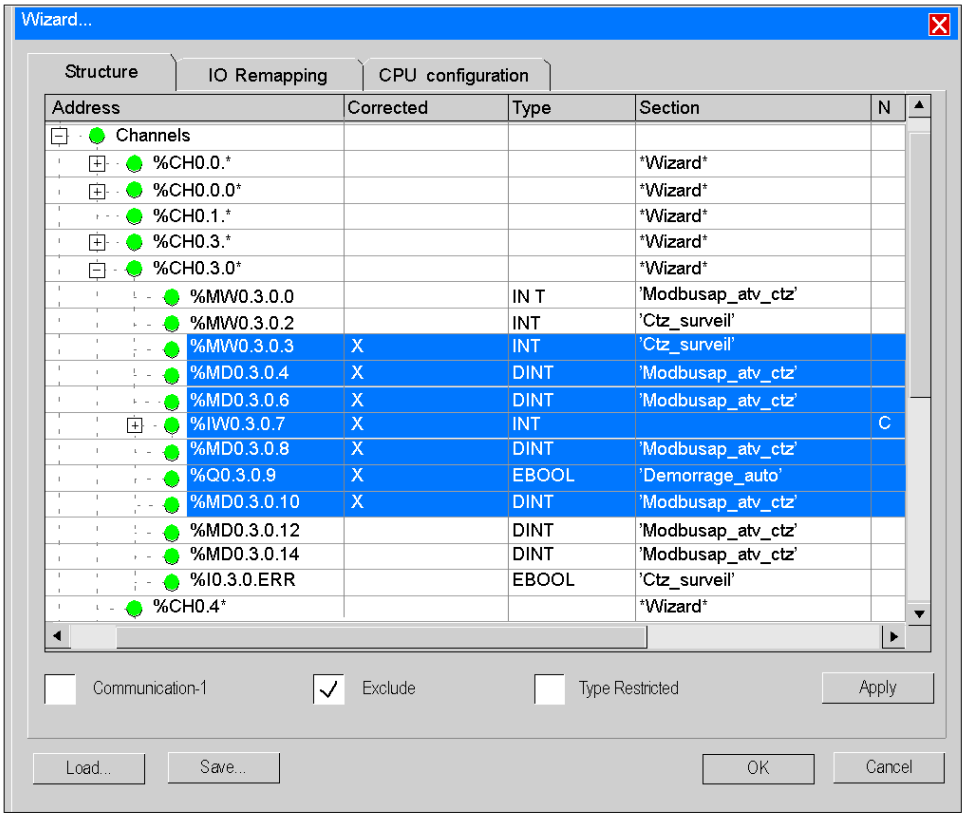
# Exclude, Communication -1, Type Restriction

## Overview

The **IO Remapping** tab provides additional features to ease special remapping operations (**Communication -1**, **Exclude**, **Type Restricted**).

These operations can be executed by checking the appropriate checkboxes below the remapping table.

## Representation



### Communication -1

Premium or Modicon M340 memory is indexed starting with 0, whereas Compact or Quantum memory indexing starts with 1.

If an application for the Compact or Quantum family is converted partially into a Premium or Modicon M340 family application, and if this PLC must communicate through the Modbus protocol with another PLC, it will be necessary to adapt the addresses during the conversion to recover consistent exchanges with other PLCs.

The communication software introduces this offset.

To subtract a 1 of all selected addresses, which must be determined to be Modbus communication related, activate the **Communication -1** checkbox.

### Exclude

To exclude I/O objects from conversion proceed as described below.

| Step | Action |
|---|---|
| 1 | Select the I/O objects in the remapping table you want to exclude from conversion. |
| 2 | Activate the **Exclude** checkbox. |
| 3 | Click the **Apply** button.<br>**Result:**<br>An X character is displayed in the **Corrected** column of the I/O objects excluded from conversion. |

### Type Restricted

In some cases not all information of a channel has to be remapped commonly.

To remap variables that match to one type within a selection only, activate the **Type Restricted** checkbox. See example below.

Type restriction applies to all kind of directly addressed variables.

%M internal variables

| Variable | Type |
|---|---|
| %M | bit |
| %MW | word |
| %MD (for Premium only) | double word |
| %MF (for Premium only) | real (floating point) |

%K constants

| Variable | Type |
|---|---|
| %KW | word constant |
| %KD (for Premium only) | double word constant |
| %KF (for Premium only) | real (floating point) constant |

%I inputs

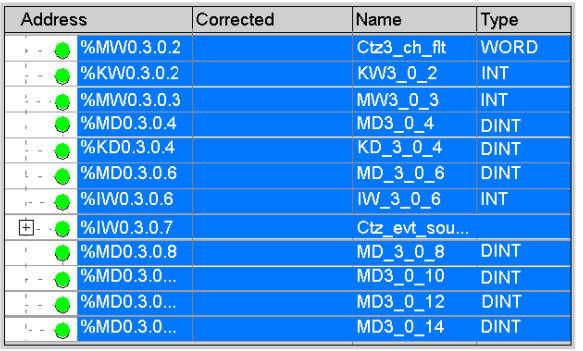
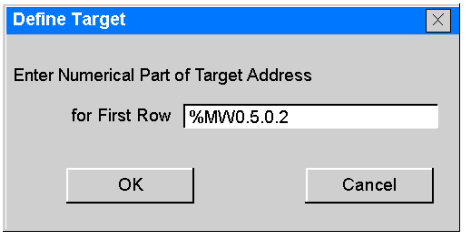
| Variable | Type |
|---|---|
| %I | BOOL or EBOOL type bit |
| %IW | INT type word |
| %ID (for Premium only) | DINT type double word |
| %IF (for Premium only) | REAL type real (floating point) |

%Q outputs

| Variable | Type |
|---|---|
| %Q | EBOOL type bit |
| %QW | INT type word |
| %QD (for Premium only) | DINT type double word |
| %QF (for Premium only) | REAL type real (floating point) |

## Type Restricted Example

For example the **Type Restricted** feature allows to remap all %MW variables at once to a module in the target family.

| Step | Action |
|---|---|
| 1 | Select the variables to be remapped.<br>  <br> |
| 2 | Activate the **Type Restricted** checkbox. |
| 3 | Select **Define Target** from the context menu and enter the new address for the first variable to be remapped.<br>  <br> |

| Step | Action |
|---|---|
| 4 | All addresses, which have the same type (e.g. %MW) as the first one selected, are remapped. |

| Address | Corrected | Name | Type |
|---|---|---|---|
| %MW0.3.0.2 | %MW0.5.0.2 | Ctz3_ch_flt | WORD |
| %KW0.3.0.2 | | KW3_0_2 | INT |
| %MW0.3.0.3 | %MW0.5.0.3 | MW3_0_3 | INT |
| %MD0.3.0.4 | | MD3_0_4 | DINT |
| %KD0.3.0.4 | | KD_3_0_4 | DINT |
| %MD0.3.0.6 | | MD_3_0_6 | DINT |
| %IW0.3.0.6 | | IW_3_0_6 | INT |
| %IW0.3.0.7 | | Ctz_evt_sou... | |
| %MD0.3.0.8 | | MD_3_0_8 | DINT |
| %MD0.3.0... | | MD3_0_10 | DINT |
| %MD0.3.0... | | MD3_0_12 | DINT |
| %MD0.3.0... | | MD3_0_14 | DINT |

# Section 53.4
# CPU Configuration Tab of the Conversion Wizard

## Introduction

### Overview

The **CPU Configuration** tab of the conversion wizard allows to modify the amount of used memory in the CPU.

The merge of corrected values and original values is put directly to the opened application.

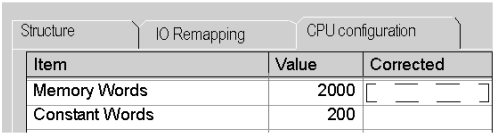
If there is a corrected value, it is used; otherwise the original value is taken.

The limits are check after converting the application, during the analysis of the application in Unity Pro only.

The **CPU Configuration** tab is available for all PLC families with the exception that the input for **Constant Words** is not available for the Quantum family.

### CPU Configuration Tab

To modify the values type the new values directly into the respective rows of the **Corrected** column.



# Appendices

## Subject of this Part

This part presents the hyperlink functions, the ergonomics, and the shortcuts of Unity Pro.

## What Is in This Appendix?

The appendix contains the following chapters:


| Chapter | Chapter Name | Page |
|---|---|---|
| A | Hyperlinks in Unity Pro | 1817 |
| B | User Interface | 1885 |
| C | Keyboard Shortcuts | 1917 |

# Appendix A
# Hyperlinks in Unity Pro

## Subject of this Chapter

This chapter describes how to use hyperlinks (hypertext) in Unity Pro.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| A.1 | General | 1818 |
| A.2 | Principles | 1823 |
| A.3 | Hyperlinks in the project browser | 1834 |
| A.4 | Hyperlink in the data editor | 1841 |
| A.5 | Hyperlinks in the DFB type attributes | 1849 |
| A.6 | Hyperlinks in animation tables | 1860 |
| A.7 | Hyperlinks in runtime screens | 1867 |
| A.8 | Hyperlinks in the documentation folder | 1874 |
| A.9 | Hyperlinks and drag-and-drop | 1881 |

# Section A.1
# General

## Object of this sub-section

This sub-section provides general information about hyperlinks.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Introduction to Hyperlinks | 1819 |
| Representation and opening of a hyperlink | 1822 |

# Introduction to Hyperlinks

## General

You can use the hyperlink function available in the Unity Pro software to create links between your project and external documents.

Hyperlinks are shortcuts to such documents as:

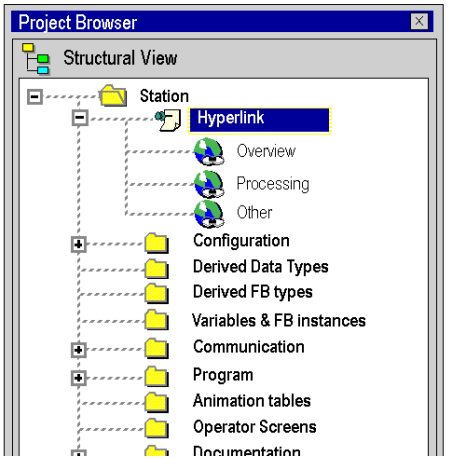
- installation and wiring diagrams,
- help screens,
- miscellaneous notes,
- images,
- web pages,
- etc.

For example, in an operator screen, you can associate an alarm that is displayed on a variable controller to a hyperlink that opens a document explaining the alarm's characteristics.

## Hyperlinks and Project Browser

You can create hyperlinks in the project browser. They will be managed like shortcut sheets in a directory.

Example:




**NOTE:** To drop a hyperlink in a blank user directory, you must press the ‘shift’ key during drag-and-drop.

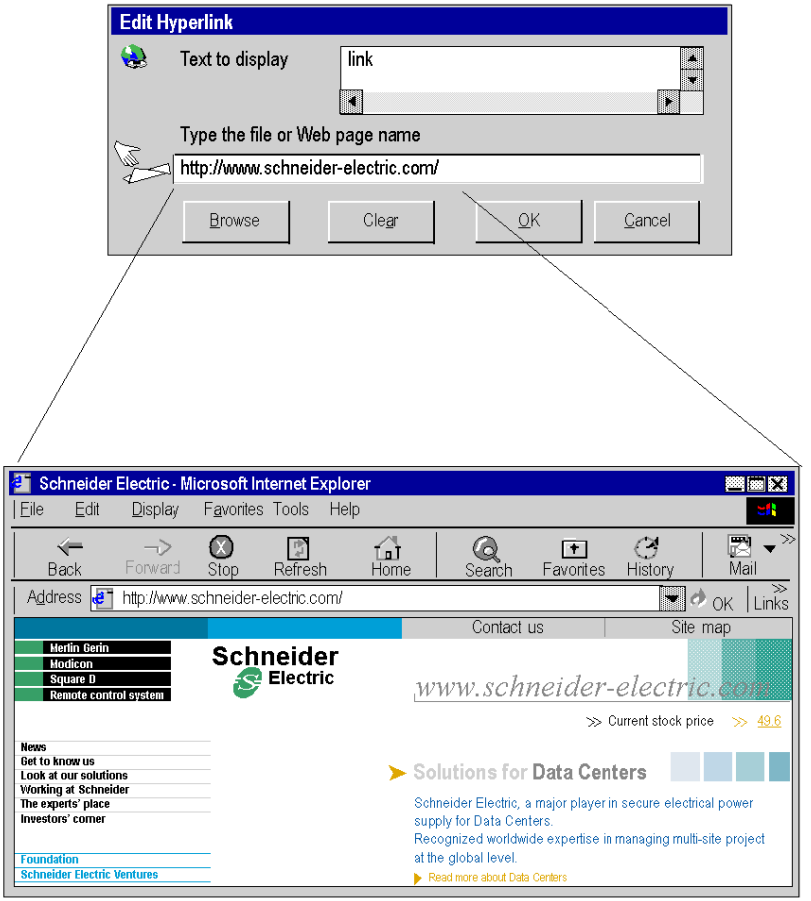
## Hyperlinks and Comments

You can also create hyperlinks from a comment field. A hyperlink in a comment is used to extend the comment to other data outside the controller project which can take on all forms to represent the information.

In the Unity Pro functions below, you can create a hyperlink in a comment belonging to:

- data editor variables,
- the animation tables,
- the operator screens,
- documentation.

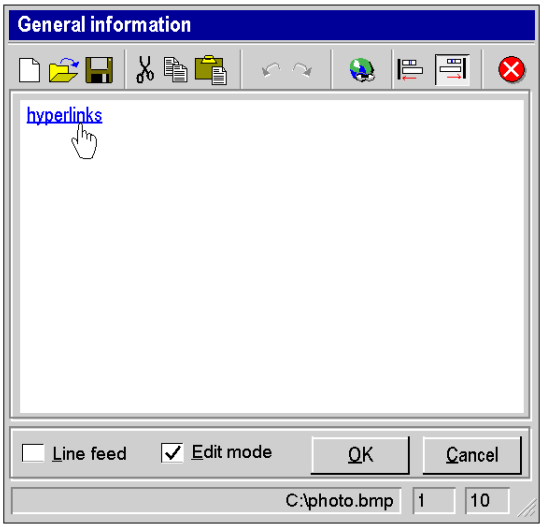
Example:


Edit Hyperlink
Text to display
link
Type the file or Web page name
http://www.schneider-electric.com/
Browse
Clear
OK
Cancel
Schneider Electric - Microsoft Internet Explorer
File Edit Display Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address http://www.schneider-electric.com/
OK Links
Contact us
Site map
Merlin Gerin
Modicon
Square D
Remote control system
Schneider Electric
www.schneider-electric.com
Current stock price 49.6
News
Get to know us
Look at our solutions
Working at Schneider
The experts' place
Investors' corner
Foundation
Schneider Electric Ventures
Solutions for Data Centers
Schneider Electric, a major player in secure electrical power supply for Data Centers.
Recognized worldwide expertise in managing multi-site project at the global level.
Read more about Data Centers

# Representation and opening of a hyperlink

## Representation

A hyperlink is represented by an underlined text in blue. When you move the mouse cursor over the hyperlink, it changes into a hand.




## Opening a link

Right-click on the underlined part to open the linked document.

# Section A.2
## Principles

### Object of this sub-section

This sub-section presents the main operate-modes and characteristics of hyperlinks.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Definitions | 1824 |
| Overview of operate-modes | 1825 |
| 1-step mode | 1826 |
| 2-step mode | 1827 |
| Description of the Hyperlink Comment Window | 1829 |
| Properties | 1832 |
| List of barred characters | 1833 |

# Definitions

## Target document

You CANopen the target document by clicking on the hyperlink. There are no restrictions on the type of document that can be opened. The system simply has to know the document's suffix. In other words the application at the origin of the creation of the document or which can view must exists and be available on the computer.

## Hyperlink target

The hyperlink target is the tool (software) that is used to open the document. For example, if the document has a .doc extension, the hyperlink target is Microsoft Word.

The table below presents the different hyperlink targets depending on the file extensions. This list is not exhaustive.

| File extensions | Hyperlink target |
|---|---|
| .doc | Microsoft Word |
| .xls | Microsoft Excel |
| .txt | NotePad |
| .htm<br>.html<br>.asp<br>.shtml | Microsoft Internet Explorer<br>Netscape<br>WEB browser |
| .bmp<br>.jpg<br>.gif | Paint<br>Paint shop |
| .png | Microsoft Photo Editor |

# Overview of operate-modes

## Operate-modes

There are two operate-modes for creating and accessing a hyperlink:

- 1-step *(see page 1826)*mode,
- 2-step *(see page 1827)*mode.

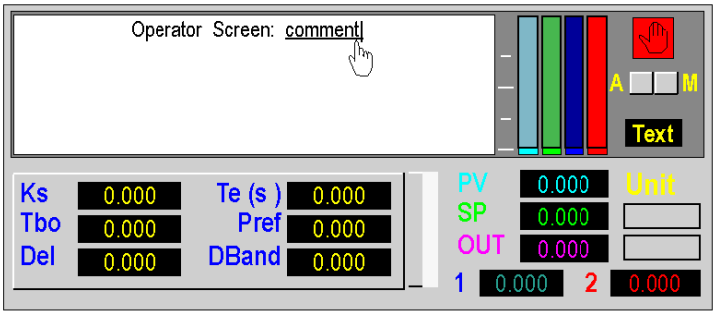
Each of these modes is applied to specific functions of the Unity Pro application.

# 1-step mode

## Principle

The 1-step mode is the mode that is available for most Unity Pro functions.

It consists of associating a hyperlink with a comment. The principle of the 1-step mode is that the target document can be accessed directly with a single click from the comment. Simply click on the underlined comment to open the linked document.




## Available functions

You can create a hyperlink in comments and objects that belong to the following functions:

- In the project browser:
  - directories and sub-directories of the browser,
  - hyperlink directory.
- In the data editor on:
  - elementary variable (EDT) comment,
  - function block (DFB/FB) instance comment,
  - derived variable (DDT) comment.
- In animation tables on:
  - variable (EDT/DDT/FB) comment.
- In operator screens on:
  - text graphical object,
  - animated text objects with a message.

# 2-step mode

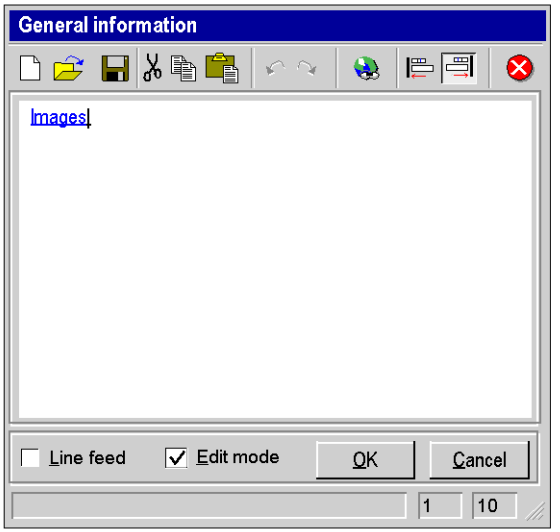
## Principle

This mode is described as 2-step because you cannot access the target document directly. An intermediary window opens (comment editor) before you can access the hyperlink.

**NOTE:** In 2-step mode, you can create several hyperlinks in the comment editor. You can therefore have different target documents with one hyperlink source

## Accessing the 2-step mode

The table below describes how to access the 2-step mode:

| Step | Action |
|---|---|
| 1 | Use the project browser to select the **General Information** sub-directory of the **Documentation** directory. |
| 2 | Select **Open** in the contextual menu.<br>**Result**: The **General Information** *(see page 1829)* window appears.<br>  |
| 3 | Click on the underlined comment in blue to open the linked document. |

## Available functions

You can create a hyperlink with the 2-step mode in comments and objects that belong to the following functions:

- In the project folder:
  - general information about the folder.
- In DFB type attributes.

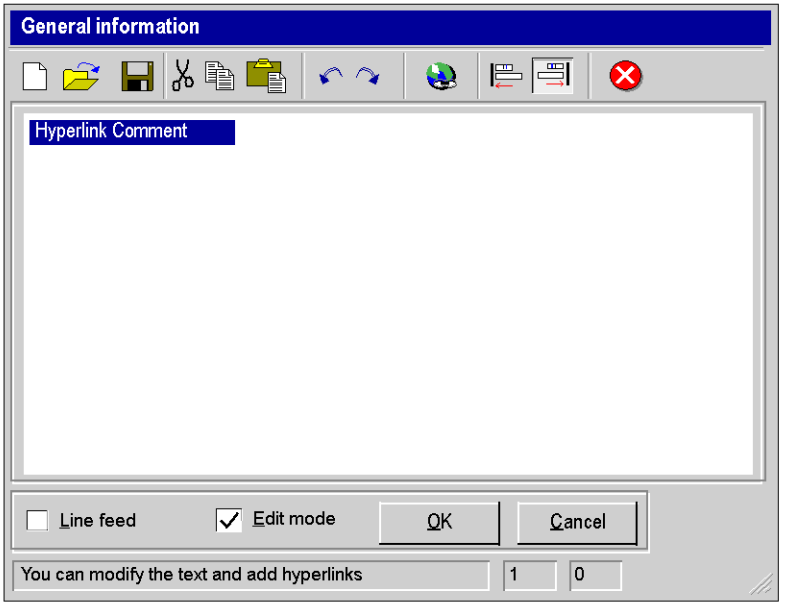
# Description of the Hyperlink Comment Window

## Illustration

In 2-step mode, the **General Information** or **DFB type description** window is used to enter the comment on which the hyperlink will be applied.

## Illustration

The functions of the icons and elements that make up the window are described in the paragraphs below.




## Toolbar

The toolbar is made up of different icons.

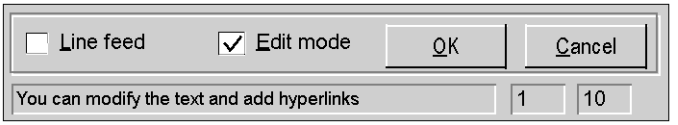


The table below presents the various functions of the icons.

| Icon | Shortcut | Description |
|---|---|---|
|  | Ctrl + N | Creates a new comment by deleting the comment that has already been written. |
|  | Ctrl + O | Opens the dialog box for opening a .htm file to retrieve a hyperlink text saved on the hard disk in a HTML page outside the controller project.<br>**Note**: You can in this way use a hyperlink that has already been created. |
|  | Ctrl + S | Saves the comment and the hyperlink(s) to the hard disk in a HTML page outside the controller project.<br>**Note**: You can in this way save a hyperlink to use again later on. This function is optional. When you save the controller project, the hyperlink comments are also saved in the .STU project. You therefore do not need to save a specific hyperlink comment in an .html page, except if you wish to use it again later on.<br>The hyperlink comments are also saved in the STA archive files. |
|  | Ctrl + X | Cuts the selected text. |
|  | Ctrl + C | Copies the selected text. |
|  | Ctrl + V | Pastes the cut or copied text. |
|  | Ctrl + Z | Undoes the last action. |
|  | | Repeats the action made before it was canceled. |
|  | Ctrl + K | Opens the **Insert hyperlink** dialog box. |
|  | | Aligns the text of the status bar to the left. |
|  | | Aligns the text of the status bar to the right.<br>**Note**: When you move the mouse over a hyperlink without clicking on it, the full path name of the document associated with the hyperlink is displayed in the status bar. However, if the path name is too long, the status bar will not be large enough to read the name of the document. The right alignment function can be useful if you want to view the name of the document associated with a hyperlink. |
|  | | Cancels and closes the window. |

## Status Bar

The status bar is made up of different elements.




The table below presents the various functions of these elements.

| Element | Description |
|---|---|
| Line feed | Automatic line skip with a long comment.<br>Check to validate this function. |
| Edit mode | Allows the comment to be entered. If you don't check this box, you will not be able to write or modify the comment.<br>Check to validate this function. |
| OK | Confirm the editing of the comment. |
| Cancel | Cancel. |
| You can modify the text and add hyperlinks<br>Explains the description of checked boxes. | |
| 1 0 | Shows the position of the cursor in the comment entry field (row - column). |

## Properties

### Storage

Comments with hyperlinks, but not the linked documents, are stored in the controller. The linked documents may be stored on the hard disk of the programming station, on the Internet, on an operator terminal or other medium.

### Operating modes

In the variable editor and folder, you can access the hyperlink in online and offline modes using the PLC.

You can modify a link while connected to the controller.

In the operator screens, you can create or edit the hyperlink in offline mode and run it in online mode.

### Display

When you open the target document, the application at the origin of the creation of the document or that is used to view the document becomes active. The Unity Pro window is however not active. The Unity Pro application continues to run as a background task.

Your actions on the keyboard will be directed to the document that you have just opened and not to the Unity Pro application.

To make the Unity Pro window active again, click on the window or on the icon in the taskbar.

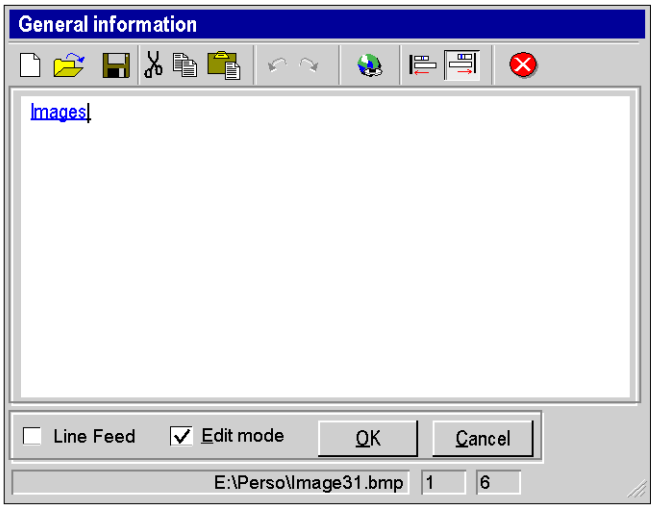
### Drag / Drop

You can Drag & Drop *(see page 1881)* icons between the file explorer and the hyperlink insertion windows. You can, for example, insert the full path of the target document.

# List of barred characters

## Principle

When you enter a comment for a hyperlink in the **General Information** or **DFB type description**, window, the characters that are allowed are 8-bit characters of the ANSI alphabet.




You can write them in the language of your choice within the limits of the alphabet allowed.

## Barred characters

16-bit UNICODE characters outside the ANSI range are not allowed.

# Section A.3
## Hyperlinks in the project browser

### Object of this sub-section

This sub-section presents the operate-mode for creating, selecting and deleting a hyperlink in the project browser.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Creating a hyperlink in the project browser | 1835 |
| Editing the properties of a hyperlink in the project browser | 1837 |
| Opening a hyperlink from the project browser | 1839 |
| Deleting a hyperlink in the project browser | 1840 |

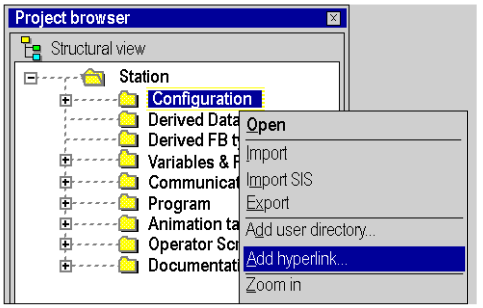
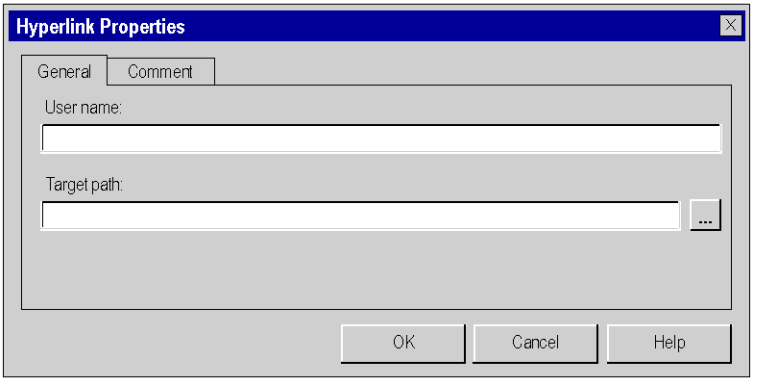
# Creating a hyperlink in the project browser

## At a Glance

You can create hyperlinks in all the directories and sub-directories of the project browser. For example, you can create a hyperlink in the **Program** directory. You can also create another hyperlink in the **Sections** sub-directory of the **MAST task** directory.

## Procedure

The table below describes the procedure for creating a hyperlink in the project browser using an existing directory.

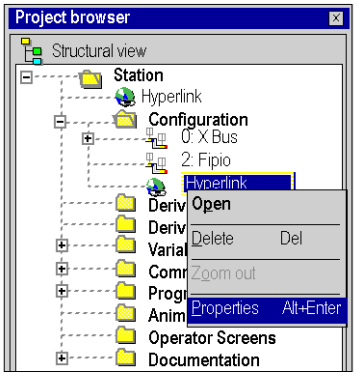
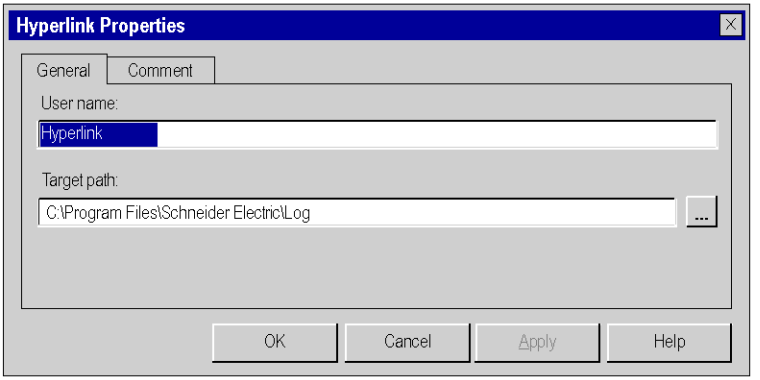
| Step | Action |
|---|---|
| 1 | Select the desired directory in the **Project browser**: |
| 2 | Right-click the selected directory.<br>**Result**: The contextual menu appears.<br>  |
| 3 | Click on **Add Hyperlink** in the contextual menu.<br>**Result**: The **Hyperlink Properties** dialog box appears.<br>  |

| Step | Action |
|---|---|
| 4 | Enter the name of the hyperlink in the **User Name** field. |
| 5 | In the **Target Path** field:<br><br>● use the  button to choose the disk and/or the directory to select the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● enter the Internet address (example: http://www.schneider-electric.com). |
| 6 | If you want to add a comment, select the **Comment** tab and enter your text in the entry field. |
| 7 | Click **OK** to confirm your choice. |

# Editing the properties of a hyperlink in the project browser

## Procedure

The table below describes the procedure for editing the properties of a hyperlink in the project browser.

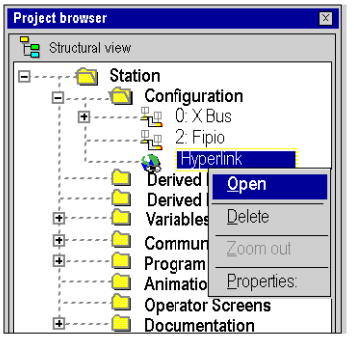
| Step | Action |
|---|---|
| 1 | Select the hyperlink to be edited in the **Project browser**. |
| 2 | Right-click the selected hyperlink.<br>**Result**: The contextual menu appears.<br>  |
| 3 | Click on **Properties** in the contextual menu.<br>**Result**: The **Hyperlink Properties** dialog box appears.<br>  |
| 4 | Modify the name of the hyperlink in the **User Name** field. |

| Step | Action |
|---|---|
| 5 | In the **Target Path** field:<br>● use the ... button to choose the disk and/or the directory to select the target document (example: c:\Program Files\Schneider Electric\log\example.doc) or<br>● Change the Internet address (example: http://www.schneider-electric.com). |
| 6 | If you want to add a comment, select the **Comment** tab and enter your text in the entry field. |
| 7 | Click **OK** to confirm the changes or **Cancel** to cancel. |

# Opening a hyperlink from the project browser

## Procedure

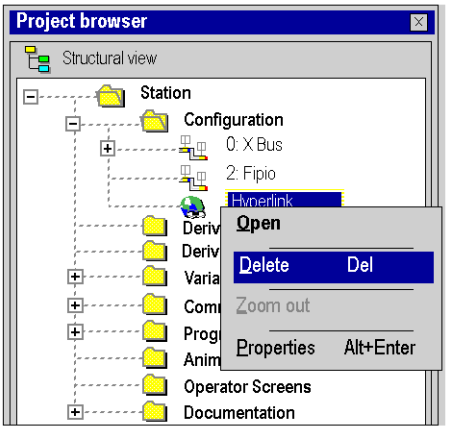
The table below describes the procedure for opening a hyperlink from the project browser.

| Step | Action | |
|---|---|---|
| 1 | Select the hyperlink in the **Project browser**: | |
| 2 | Double click on the hyperlink represented by an icon with the name given to it when it was created.<br>example:  | Right-click the selected hyperlink.<br>**Result**: The contextual menu appears.<br>  |
| 3 | | Select **Open** in the contextual menu. |
| 4 | **Result**: The external document is launched. | |

# Deleting a hyperlink in the project browser

## Procedure

The table below describes the procedure for deleting a hyperlink in the project browser.

| Step | Action |
|---|---|
| 1 | Select the hyperlink to be deleted in the **Project browser**. |
| 2 | Right-click the selected hyperlink.<br>**Result**: The contextual menu appears.<br>  |
| 3 | Click **Delete** in the contextual menu.<br>**Result**: The hyperlink disappears. |

# Section A.4
## Hyperlink in the data editor

### Object of this sub-section

This sub-section presents the operate-mode for creating, selecting, editing and deleting a hyperlink in the data editor.

### What Is in This Section?

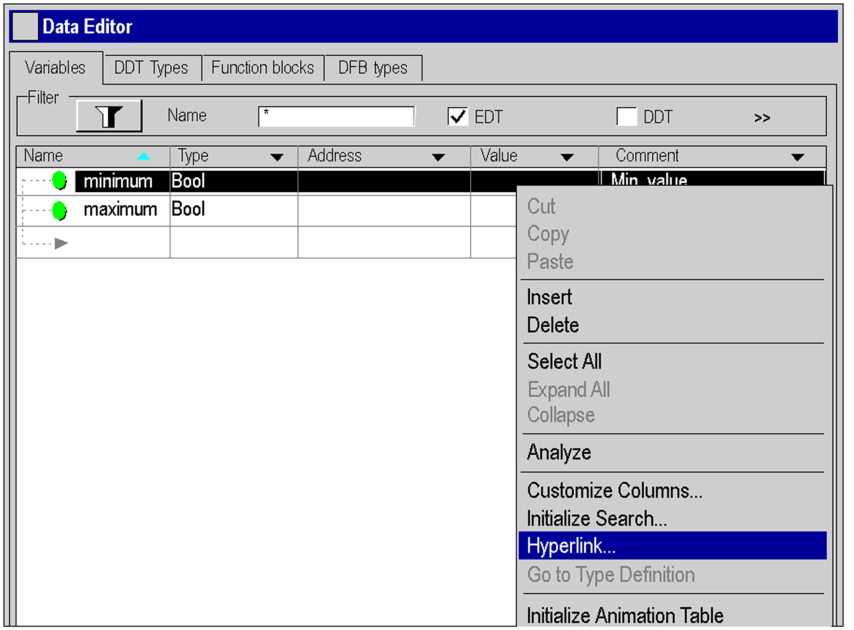
This section contains the following topics:

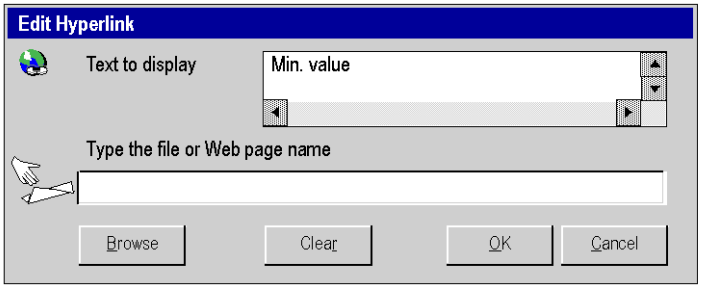

| Topic | Page |
|---|---|
| Creating a hyperlink in the data editor | 1842 |
| Editing a hyperlink in the data editor | 1844 |
| Selecting a hyperlink in the data editor | 1846 |
| Deleting a hyperlink in the data editor | 1847 |

## Creating a hyperlink in the data editor

### Procedure

The table below describes the procedure for creating a hyperlink in the data editor with a 1-step mode.

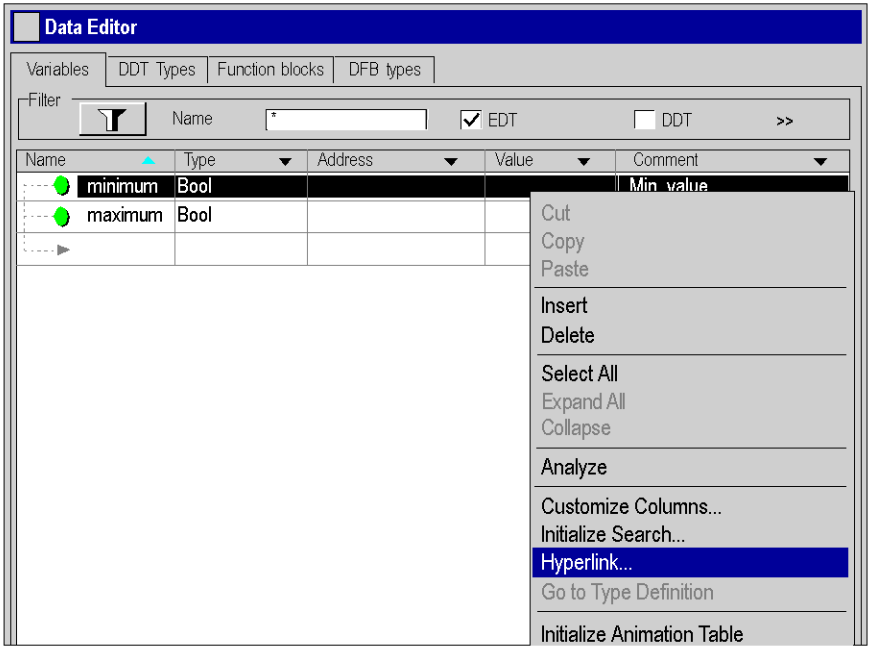
| Step | Action |
|---|---|
| 1 | Open **the data editor**. |
| 2 | Select the comment where you wish to create a hyperlink in the **Comment** column of the **Variables** tab. |
| 3 | Right-click the selected comment.<br>**Result**: The contextual menu appears.<br>  |

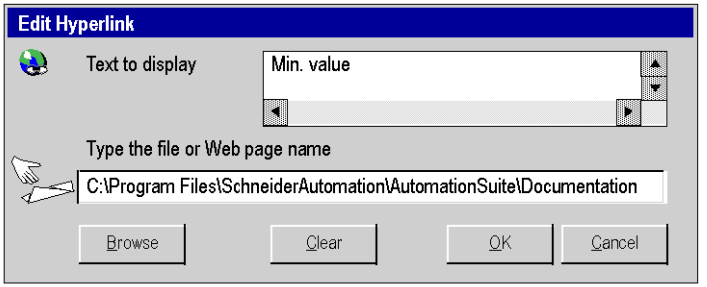
| Step | Action |
|---|---|
| 4 | Click on **Hyperlink** in the contextual menu.<br>**Result**: The **Edit Hyperlink** dialog box appears.<br>  |
| 5 | The selected comment appears in the **Text to display** field.<br>In the **Enter name of file or Web page** field:<br>● click  to select the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● enter the Internet address (example: http://www.schneider-electric.com). |
| 6 | Click **OK** to confirm your choice. |

## Editing a hyperlink in the data editor

### Procedure

The table below describes the procedure for modifying the properties of a hyperlink in the data editor.

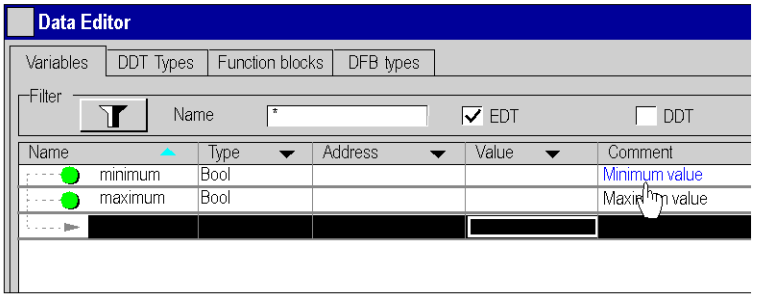
| Step | Action |
|---|---|
| 1 | Open the **data editor**. |
| 2 | Select the hyperlink comment to be edited in the **Comment** column of the **Variables** tab. |
| 3 | Right-click the selected comment.<br>**Result**: The contextual menu appears.<br>  |

| Step | Action |
|---|---|
| 4 | Click on **Hyperlink** in the contextual menu.<br>**Result**: The **Edit Hyperlink** dialog box appears.<br>  |
| 5 | The selected comment appears in the **Text to display** field.<br>In the **Type the file or Web page name** field:<br>● click  to select the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● change the Internet address (example: http://www.schneider-electric.com). |
| 6 | Click **OK** to confirm the changes or **Cancel** to cancel. |

## Selecting a hyperlink in the data editor

### Procedure

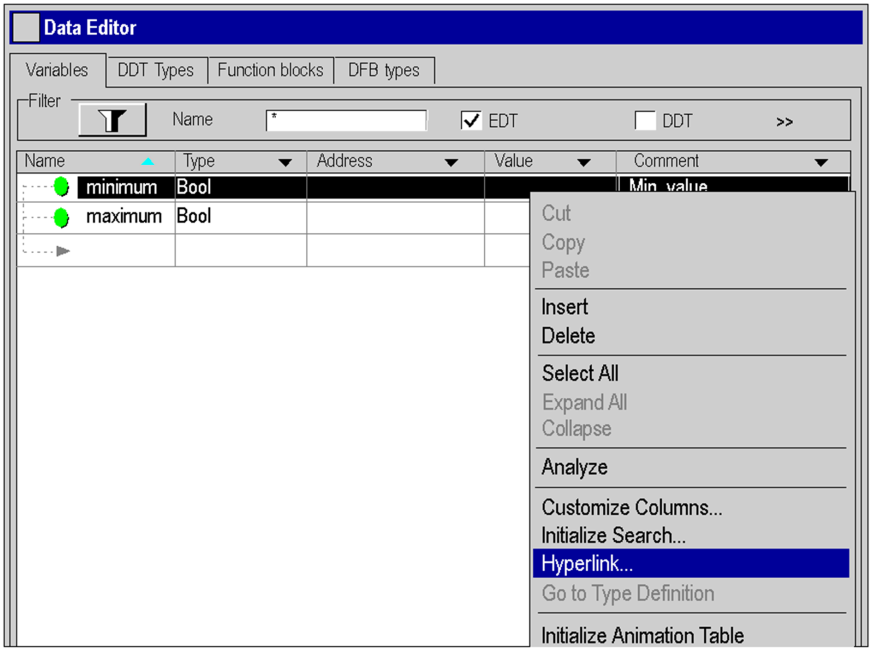
The table below describes the procedure for opening a hyperlink in the data editor.

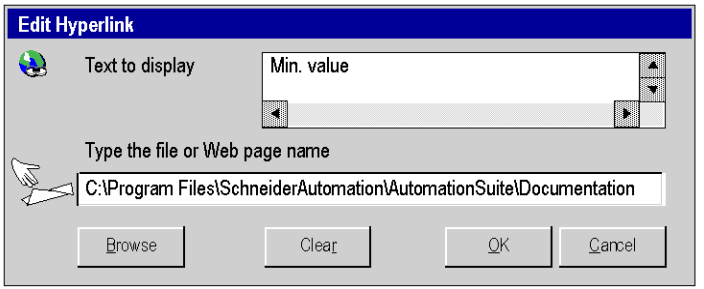
| Step | Action |
|---|---|
| 1 | Open **the data editor**. |
| 2 | In the **Comment** column of the **Variables** tab, click on the required hyperlink (text in blue/mouse cursor changes to a hand).<br>  |

# Deleting a hyperlink in the data editor

## Procedure

The table below describes the procedure for deleting a hyperlink in the data editor.

| Step | Action |
|---|---|
| 1 | Open the **data editor**. |
| 2 | Select the comment to be deleted in the **Comment** column of the **Variables** tab. |
| 3 | Right-click the selected comment.<br>**Result**: The contextual menu appears.<br>  |

| Step | Action |
|---|---|
| 4 | Click on **Hyperlink** in the contextual menu.<br>**Result**: The **Edit Hyperlink** dialog box appears.<br>  |
| 5 | Click **Delete**: the text in the **Type the file or Web page name** field is deleted. |
| 6 | Click **OK** to confirm your choice. The hyperlink is deleted. |

# Section A.5
## Hyperlinks in the DFB type attributes

### Object of this sub-section

This sub-section presents the operate-mode for creating, editing and deleting a hyperlink in the DFB type attributes.

### What Is in This Section?

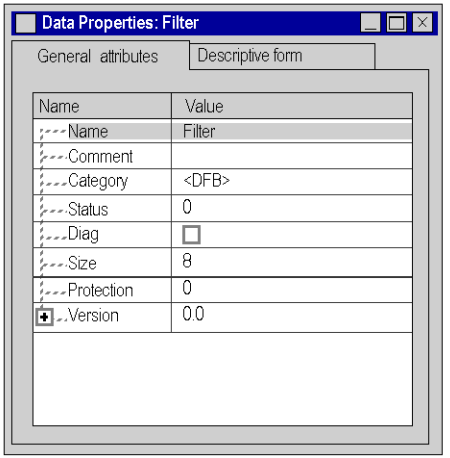
This section contains the following topics:

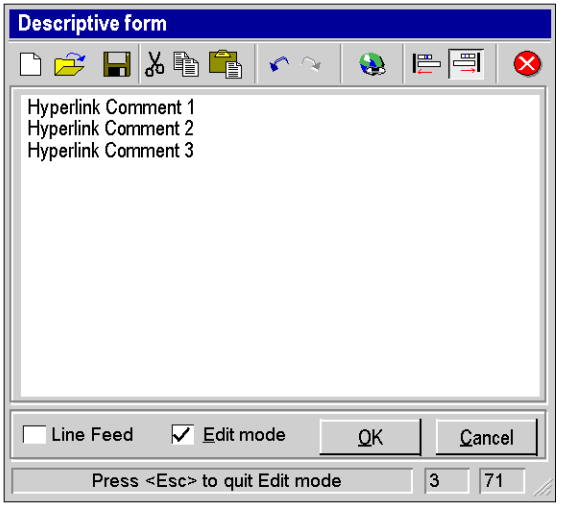
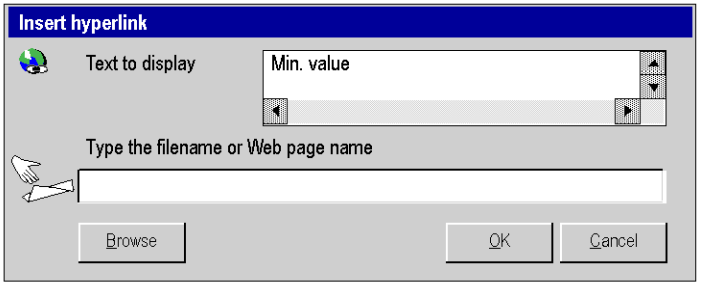

| Topic | Page |
|---|---|
| Creating a Hyperlink from DFB Type Attributes | 1850 |
| Modifying a Hyperlink from DFB Type Attributes | 1853 |
| Selecting a Hyperlink from DFB Type Attributes | 1855 |
| Deleting a Hyperlink from DFB Type Attributes | 1857 |

# Creating a Hyperlink from DFB Type Attributes

## Procedure

The table below describes the procedure for creating a hyperlink from the attributes *(see page 1268)* of a DFB type with a 2-step mode. For example, you can associate an explanatory text with the definition of the DFB type.

| Step | Action |
|---|---|
| 1 | Go to the **DFB Types** *(see page 1253)* tab of the data editor. |
| 2 | Right-click the DFB type that you want to use to create a hyperlink.<br>**Result**: the contextual menu appears. |
| 3 | Choose the **Data Properties** command.<br>**Result**: the corresponding dialog box appears on the screen.<br>  |

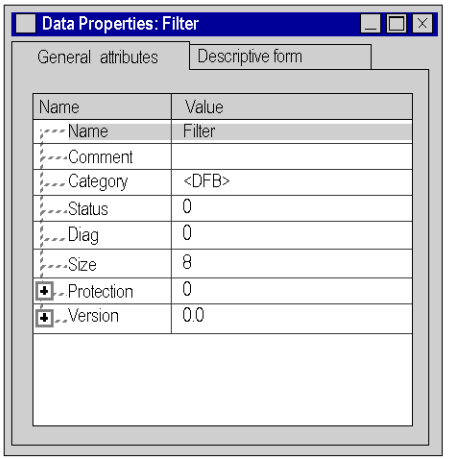
| Step | Action |
|---|---|
| 4 | Click on the **Descriptive form** tab, then on the **Edit** button.<br>**Result**: the **Descriptive form** window appears.<br>  <br>Details of window *(see page 1829)* |
| 5 | Enter a comment in the entry field. |
| 6 | Select one or more words (contiguous) from the comment you have entered and click on the  icon.<br>**Result**: the **Insert hyperlink** window appears.<br>  <br>**Note**: You can create several hyperlinks in this way by selecting different words. |

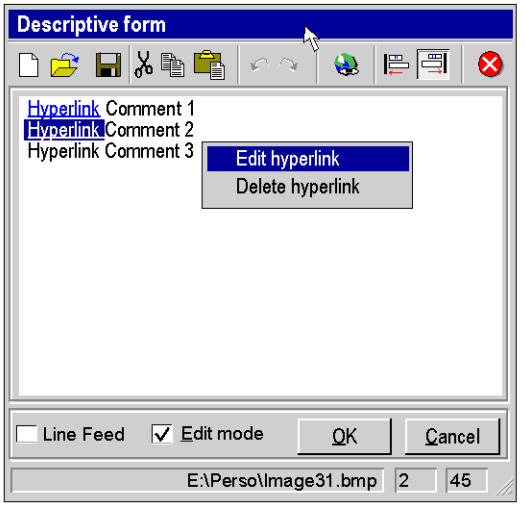
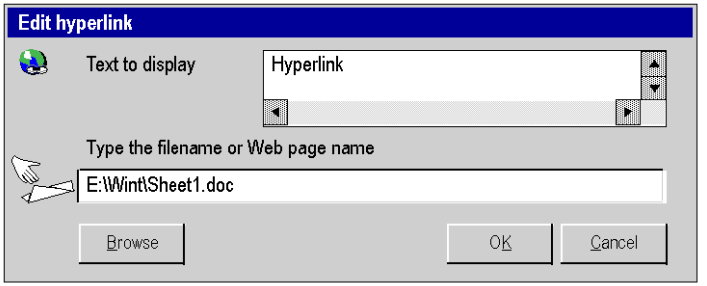
| Step | Action |
|---|---|
| 7 | The text selected in step 6 appears in the **Text to display** field.<br>In the **Type the filename or Web page name** field:<br>● click  to select the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● enter the Internet address (example: http://www.schneider-electric.com). |
| 8 | Click **OK** to confirm your choice. |

# Modifying a Hyperlink from DFB Type Attributes

## Procedure

The table below describes the procedure for editing the properties of a hyperlink in the attributes of a DFB type.

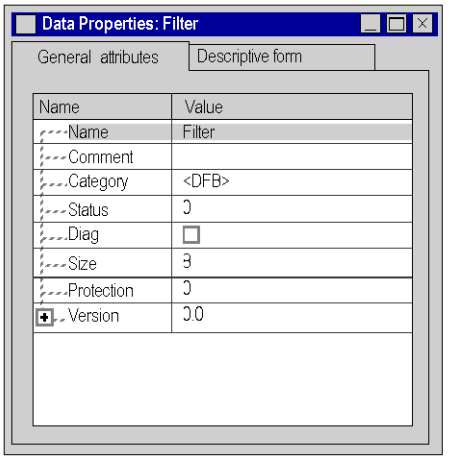
| Step | Action |
|---|---|
| 1 | Go to the **DFB Types** *(see page 1253)* tab of the data editor. |
| 2 | Right-click the DFB type whose hyperlink you wish to edit.<br>**Result**: the contextual menu appears. |
| 3 | Choose the **Data Properties** command.<br>**Result**: the corresponding dialog box appears on the screen.<br>  |
| 4 | Select the **Descriptive form** tab, and click on the **Edit** button.<br>**Result**: the **Descriptive form** window appears. |

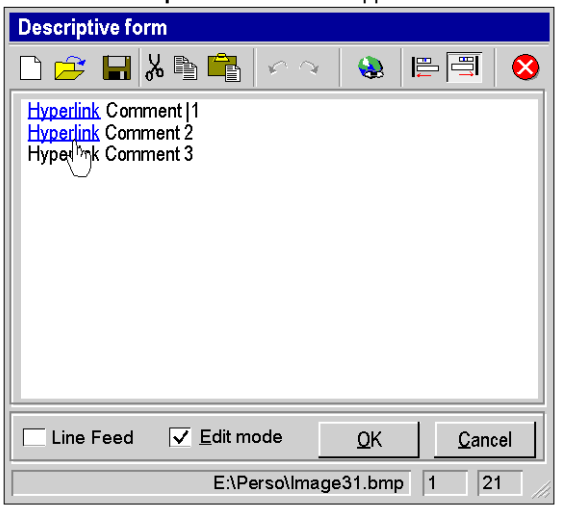
| Step | Action |
|---|---|
| 5 | Right click in the entry field, positioning the mouse on the hyperlink to edit.<br>**Result**: the contextual menu appears.<br>  |
| 6 | Click on **Edit hyperlink** in the contextual menu.<br>**Result**: the **Edit hyperlink** window appears.<br>  |
| 7 | The name of the hyperlink appears in the **Text to display** field.<br>In the **Type the filename or Web page name** field:<br>● click **Browse** to select the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● change the Internet address (example: http://www.schneider-electric.com). |
| 8 | Click **OK** to confirm your choice. |

# Selecting a Hyperlink from DFB Type Attributes

## Procedure

The table below describes the procedure for opening a hyperlink from DFB type attributes.

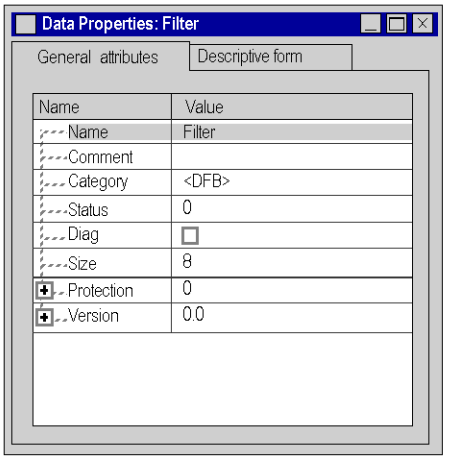
| Step | Action |
|---|---|
| 1 | Go to the **DFB Types** *(see page 1253)* tab of the data editor. |
| 2 | Right-click the DFB type whose hyperlink you wish to select.<br>**Result**: the contextual menu appears. |
| 3 | Choose the **Data Properties** command.<br>**Result**: the corresponding dialog box appears on the screen.<br>  |

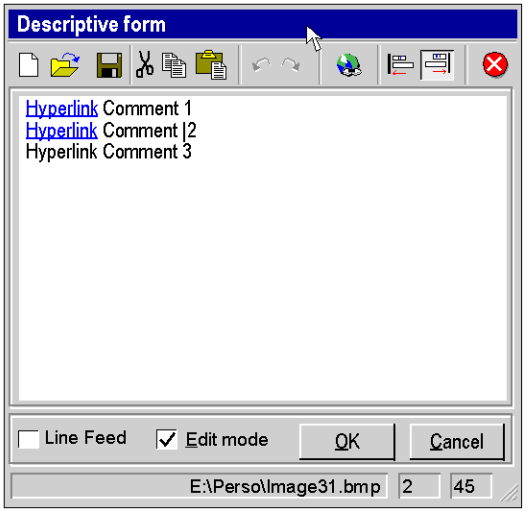
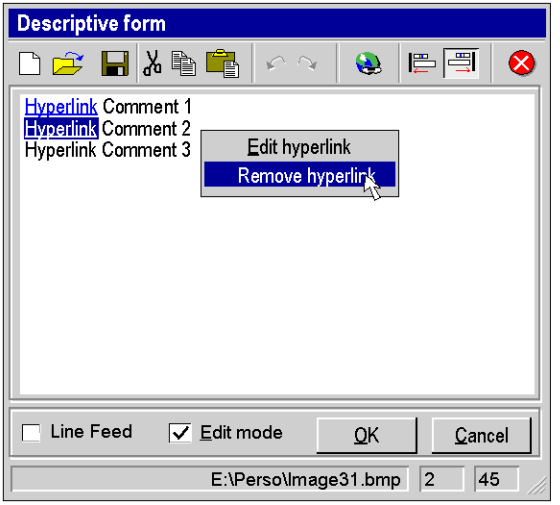
| Step | Action |
|---|---|
| 4 | From the **Descriptive form** tab, click on the **Edit** button.<br>**Result**: the **Descriptive form** window appears.<br>  |
| 5 | In the **Descriptive form** window, click on the required hyperlink (text underlined in blue/mouse cursor changes to a hand). |

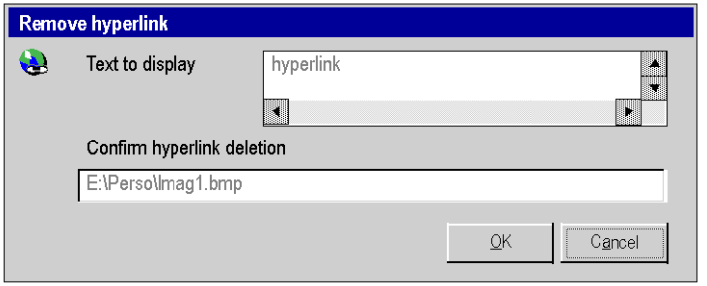
# Deleting a Hyperlink from DFB Type Attributes

## Procedure

The table below describes the procedure for deleting a hyperlink in the attributes of a DFB type.

| Step | Action |
|---|---|
| 1 | Go to the **DFB Types** *(see page 1253)* tab of the data editor. |
| 2 | Right-click the DFB type whose hyperlink you wish to delete.<br>**Result**: the contextual menu appears. |
| 3 | Choose the **Data Properties** command.<br>**Result**: the corresponding dialog box appears on the screen.<br>  |

| Step | Action |
|---|---|
| 4 | From the **Descriptive form** tab, click on the **Edit** button.<br>**Result**: the **Descriptive form** window appears.<br>  <br>Details of window *(see page 1829)* |
| 5 | Right click in the entry field, positioning the mouse on the hyperlink to be deleted.<br>**Result**: the contextual menu appears.<br>  |

| Step | Action |
|---|---|
| 6 | Click on **Remove hyperlink** in the contextual menu.<br>**Result**: the **Remove hyperlink** window appears.<br>  |
| 7 | You do not have access to the **Text to display** or **Confirm the deletion of this hyperlink** entry fields. |
| 8 | Click **OK** to confirm your choice or click **Cancel** to cancel. |

# Section A.6
## Hyperlinks in animation tables

### Object of this sub-section

This sub-section presents the operate-modes for accessing a hyperlink in the animation tables.

### What Is in This Section?

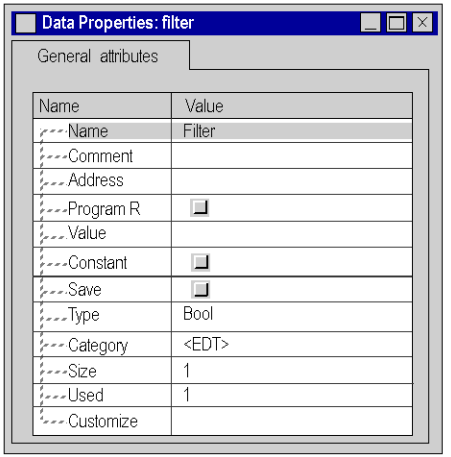
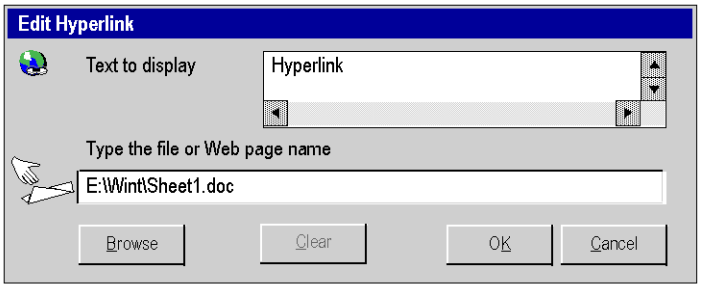
This section contains the following topics:


| Topic | Page |
|---|---|
| Creating a hyperlink in an animation table | 1861 |
| Editing a hyperlink from an animation table | 1863 |
| Selecting a hyperlink from an animation table | 1865 |
| Deleting a hyperlink from an animation table | 1866 |

# Creating a hyperlink in an animation table

## Procedure

The following table describes the procedure for creating a hyperlink in a comment for a variable in an animation table.

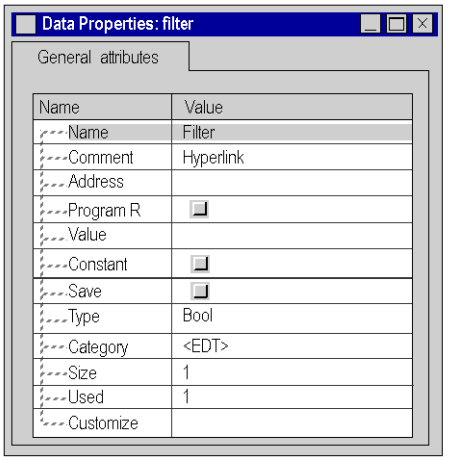
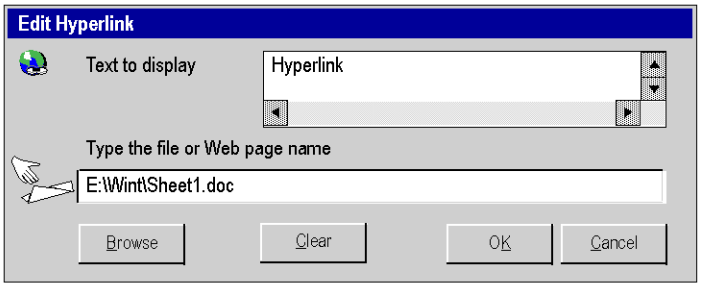
| Step | Action |
|---|---|
| 1 | Opening an animation table. |
| 2 | Select the variable where the hyperlink is to be created. |
| 3 | Select **Data Properties** from the contextual menu.<br>**Result**: The **Data Properties** dialog box appears.<br>  |
| 4 | Enter your comment in the comment line in the value column. |
| 5 | Select the comment. |
| 6 | Select **Hyperlink** in the contextual menu.<br>**Result**: The **Edit Hyperlink** window appears.<br>  |

| Step | Action |
|---|---|
| 7 | The selected comment appears in the **Text to display** field.<br>In the **Enter name of file or Web page** field:<br>● click  to select the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● enter the Internet address (example: http://www.schneider-electric.com). |
| 8 | Click **OK** to confirm your choice. |

# Editing a hyperlink from an animation table

## Procedure

The table below describes the procedure for editing the properties of a hyperlink in an animation table.

| Step | Action |
|---|---|
| 1 | Opening an animation table. |
| 2 | Select the variable where the hyperlink is located in the comment. |
| 3 | Select **Data Properties** from the contextual menu.<br>**Result**: The corresponding dialog box appears on the screen.<br>  |
| 4 | Select the comment. |
| 5 | Select **Edit Hyperlink** in the contextual menu.<br>**Result**: The **Edit Hyperlink** window appears.<br>  |

| Step | Action |
|---|---|
| 6 | The name of the hyperlink appears in the **Text to display** field.<br>In the **Type the file or Web page name** field:<br>● click **Browse** to select the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● change the Internet address (example: http://www.schneider-electric.com). |
| 7 | Click **OK** to confirm your choice. |

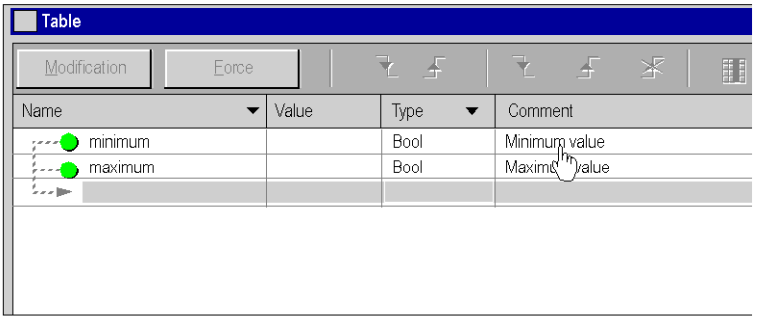
# Selecting a hyperlink from an animation table

## Principle

In an animation table, you can open a hyperlink associated with a variable comment.

## Procedure

The table below describes the procedure for opening a hyperlink in animation tables.

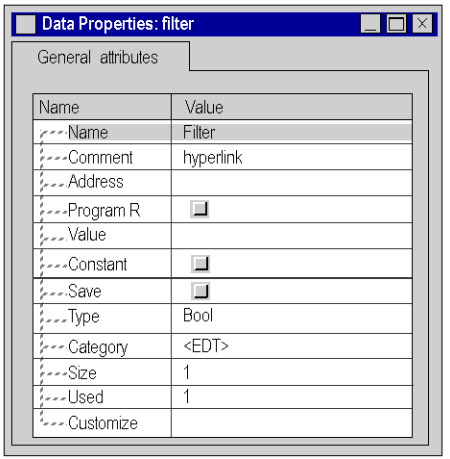
| Step | Action |
|---|---|
| 1 | Open an **animation table**. |
| 2 | In the **Comment** column, double-click on the hyperlink for the selected variable.<br>  |

# Deleting a hyperlink from an animation table

## Procedure

The following table describes the procedure for deleting a hyperlink in a comment for a variable in an animation table.

**NOTE:** This procedure is only possible if the hyperlink was created *(see page 1861)* from the Data Properties window.

| Step | Action |
|---|---|
| 1 | After creating *(see page 1861)* a hyperlink in a comment for a variable, the **Data Properties** window remains open.<br>  |
| 2 | Select the comment from which the hyperlink is to be deleted. |
| 3 | Select **Delete Hyperlink** in the contextual menu. |

# Section A.7
## Hyperlinks in runtime screens

### Object of this sub-section

This sub-section presents the operate-mode for creating, editing and deleting a hyperlink in runtime screens.

### What Is in This Section?

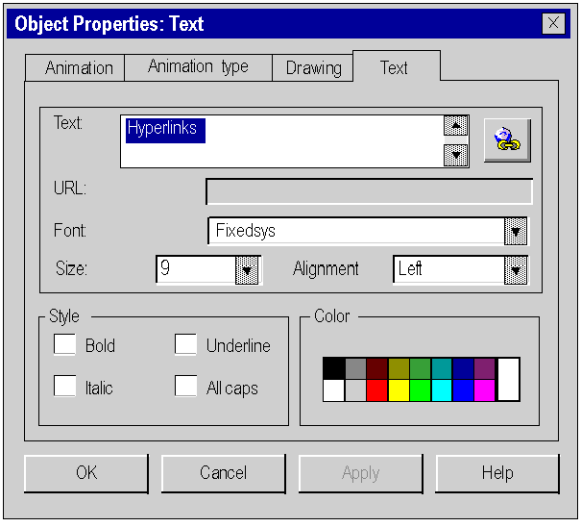
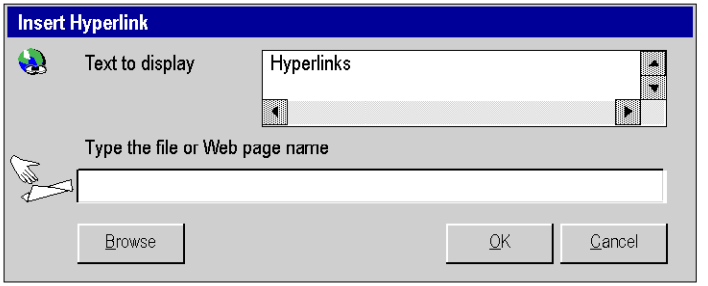
This section contains the following topics:


| Topic | Page |
|---|---|
| Creating a hyperlink from operator screens | 1868 |
| Editing a hyperlink in the operator screens | 1870 |
| Selecting a hyperlink in an operator screen | 1872 |
| Deleting a hyperlink in an operator screen | 1873 |

# Creating a hyperlink from operator screens

## Procedure

The table below describes the procedure for creating a hyperlink in operator screens.

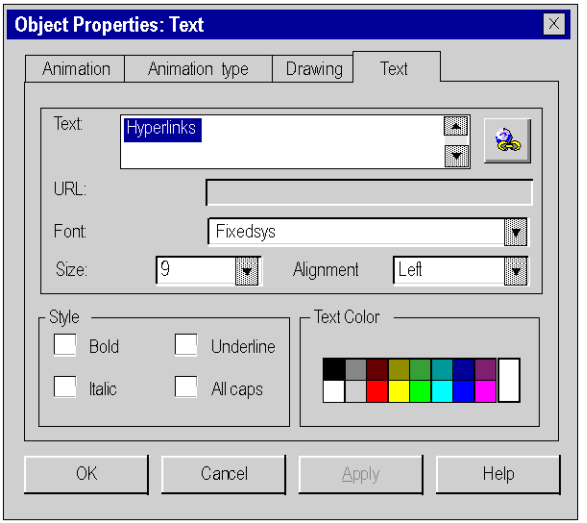
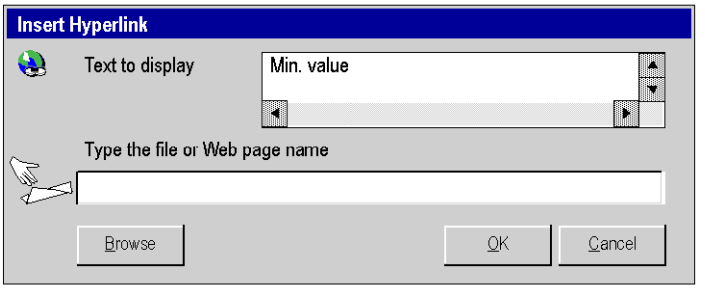
| Step | Action |
|---|---|
| 1 | Open **an operator screen**. |
| 2 | Select a **Text** object or create one. |
| 3 | Select **Properties** in the contextual menu.<br>**Result**: The following window appears.<br>  |
| 4 | Click on the icon  of the **Text** tab.<br>**Result**: The **Insert Hyperlink** window appears.<br>  |

| Step | Action |
|---|---|
| 5 | The text specified in the **Text** tab appears in the **Text to display** field.<br>In the **Type the file or Web page name** field:<br>● select the target document (C:\Program Files\Office2000\Office\Example) by using the  button or<br>● enter the Internet address (example: http://www.schneider-electric.com). |
| 6 | Click **OK** to confirm your choice. |

# Editing a hyperlink in the operator screens

## Procedure

The table below describes the procedure for editing the properties of a hyperlink in an operator screen object.

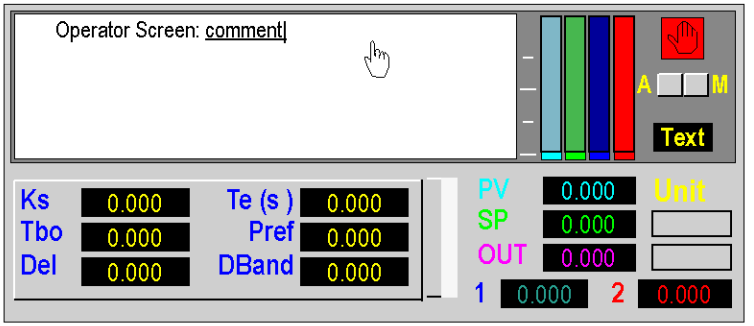
| Step | Action |
|---|---|
| 1 | Open the **operator screen**. |
| 2 | Select the **Text** object to be edited. |
| 3 | Select **Properties** in the contextual menu.<br>**Result**: The following window appears.<br>  |
| 4 | Click on the icon  of the **Text** tab.<br>**Result**: The **Insert Hyperlink** window is displayed.<br>  |

| Step | Action |
|---|---|
| 5 | The text of the **Text** tab appears in the **Text to display** field.<br>In the **Type the filename or web page name** field:<br>● use the  button to choose the disk and/or the directory to modify the target document (example: C:\Program Files\Office2000\Office\Example) or<br>● change the Internet address (example: http://www.schneider-electric.com). |
| 6 | Click **OK** to confirm the changes or **Cancel** to cancel. |

## Selecting a hyperlink in an operator screen

### Procedure

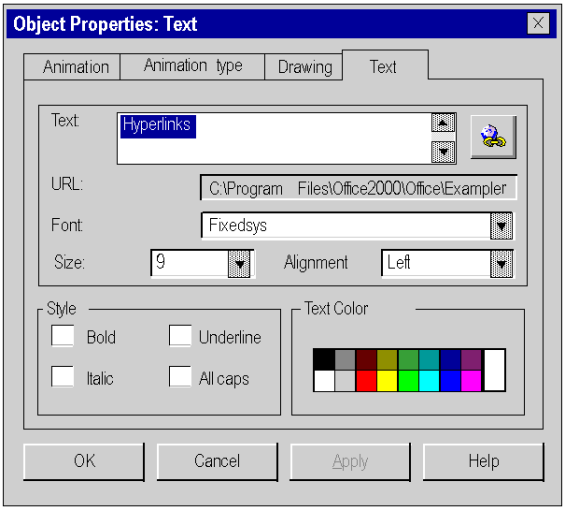
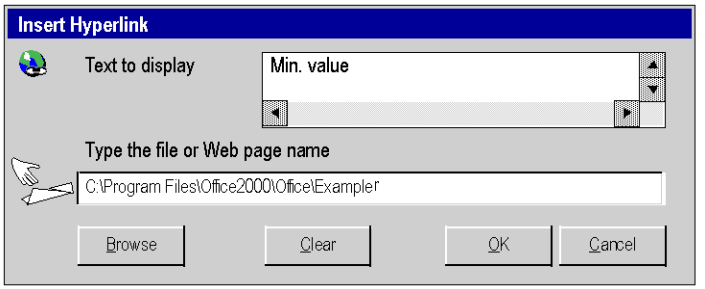
The table below describes the procedure for opening a hyperlink in an operator screen.

| Step | Action |
|---|---|
| 1 | Open **an operator screen**. |
| 2 | Click on the underlined text and the mouse cursor is changed into a thumbnail.<br>  <br>**Result**: The document associated with the hyperlink is opened. |

# Deleting a hyperlink in an operator screen

## Procedure

The table below describes the procedure for deleting a hyperlink in an operator screen.

| Step | Action |
|---|---|
| 1 | Open **the operator screen**. |
| 2 | Select the **Text** object or hyperlink to be deleted. |
| 3 | Select **Properties** in the contextual menu.<br>**Result**: The following window appears.<br>  |
| 4 | Click on the icon  of the **Text** tab.<br>**Result**: The **Insert Hyperlink** dialog box appears.<br>  |
| 5 | Click **Clear**: the text in the **Type the file or Web page name** field is cleared. |
| 6 | Click **OK** to confirm your choice. The hyperlink is deleted. |

# Section A.8
## Hyperlinks in the documentation folder

### Object of this sub-section

This sub-section presents the operate-mode for creating, editing and deleting a hyperlink in the documentation folder of a Unity Pro project.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Creating a hyperlink in the general information of the project folder | 1875 |
| Editing a hyperlink in the documentation folder | 1877 |
| Deleting a hyperlink in the documentation folder | 1879 |

# Creating a hyperlink in the general information of the project folder

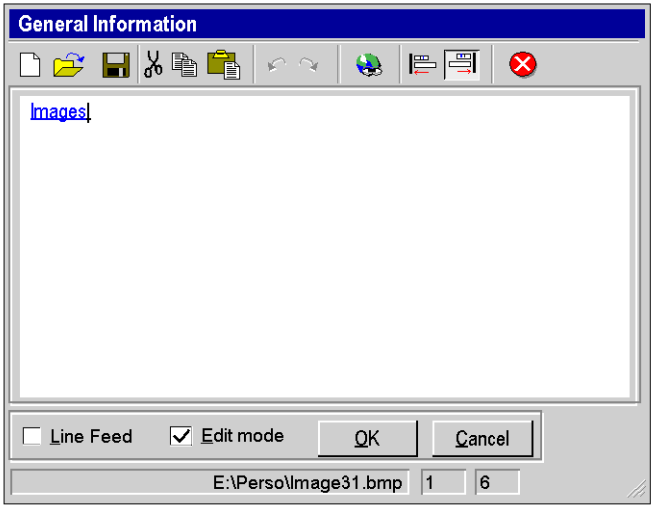
## Principle

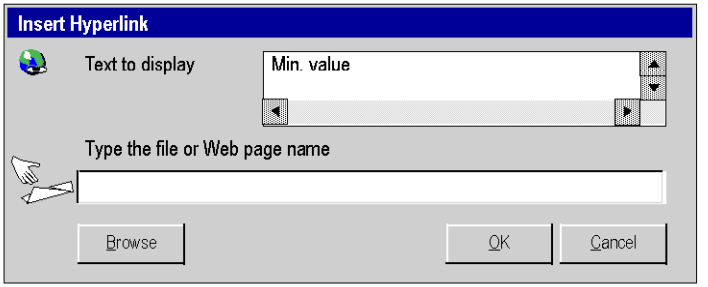
The **hyperlink** functionality is also used to make links to documents for printing in the documentation folder of a Unity Pro project. For example, they can be used to present the architecture of a project.

You can in this way create a hyperlink from **General Information** sections of the documentation folder. These are image documents (.bmp) and you can print them in the **General Information** section of the documentation folder.

## Procedure

The table below describes the procedure for creating a hyperlink in the documentation folder with a 2-step mode.

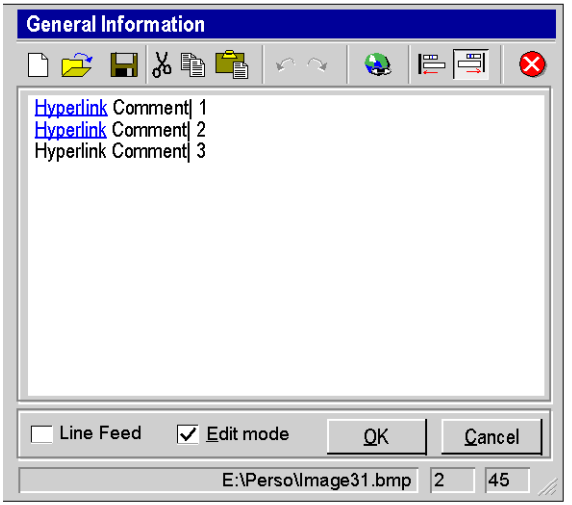
| Step | Action |
|---|---|
| 1 | Use the project browser to select the **General Information** sub-directory of the **Documentation** directory. |
| 2 | Select **Open** in the contextual menu.<br>**Result**: The **General Information** window appears.<br>  <br>Details of window *(see page 1829)* |
| 3 | Enter a comment in the entry field. |

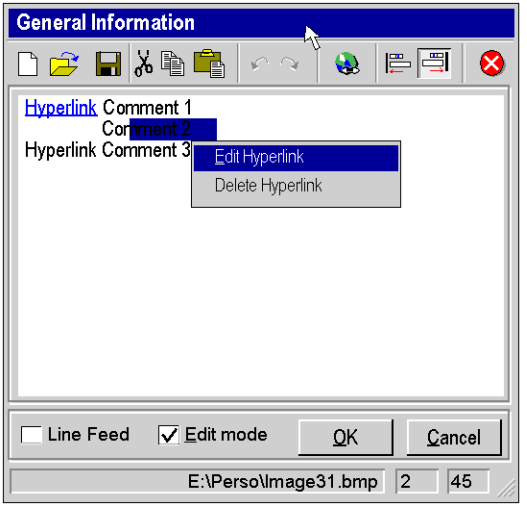
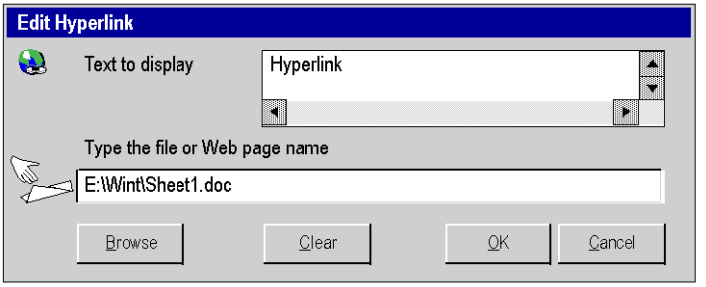
| Step | Action |
|---|---|
| 4 | Select one or more words (contiguous) from the comment you have entered and click on the icon  .<br>**Result**: The **Insert Hyperlink** window is displayed.<br>  <br>**Note**: You can create several hyperlinks in this way by selecting different words. |
| 5 | The text selected in step 4 appears in the **Text to display** field.<br>In the **Type the filename or web page name** field, use the  button to choose the disk and/or the directory to select the target document (example: C:\Images\structure1.bmp). |
| 6 | Click **OK** to confirm your choice. |

# Editing a hyperlink in the documentation folder

## Procedure

The table below describes the procedure for editing the properties of a hyperlink in the general information section of the documentation folder.

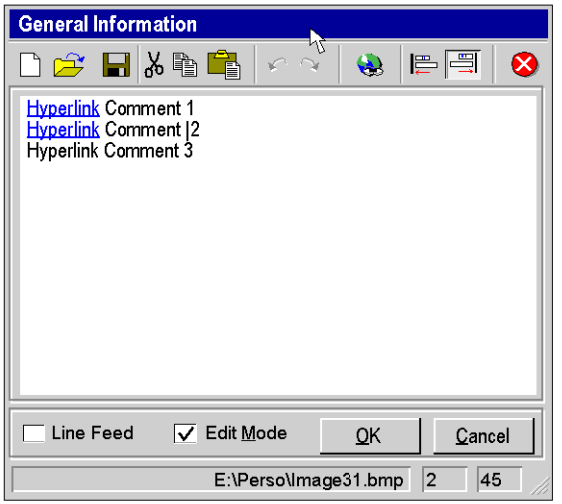
| Step | Action |
|---|---|
| 1 | Use the project browser to select the **General Information** sub-directory of the **Documentation** directory. |
| 2 | Select **Open** in the contextual menu.<br>**Result**: The **General Information** window appears.<br>  <br>Details of window *(see page 1829)* |

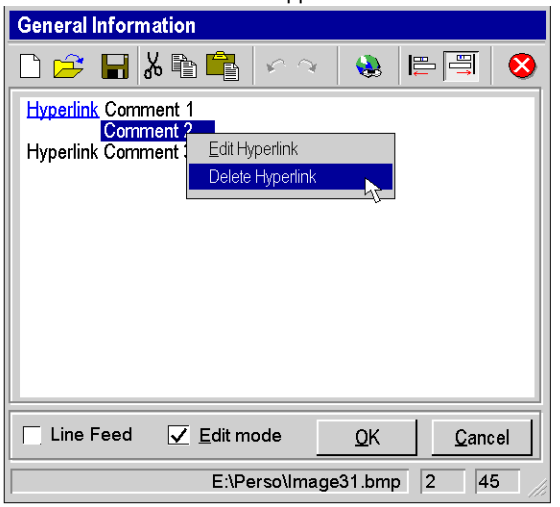
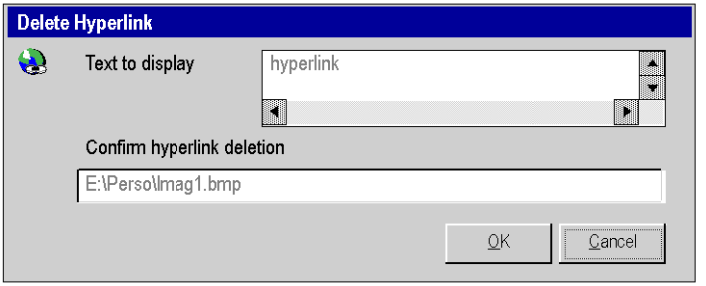
| Step | Action |
|---|---|
| 3 | Right click in the entry field, positioning the mouse on the hyperlink to edit<br>**Result**: The contextual menu appears.<br>  |
| 4 | Click on **Edit Hyperlink** in the contextual menu.<br>**Result**: The **Edit Hyperlink** window appears.<br>  |
| 5 | The name of the hyperlink appears in the **Text to display** field.<br>In the **Type the file or Web page name** field, use the  button<br>to select the target document (example: C:\Images\structure1.bmp). |
| 6 | Click **OK** to confirm your choice. |

# Deleting a hyperlink in the documentation folder

## Procedure

The table below describes the procedure for deleting a hyperlink in the general information section of the documentation folder.

| Step | Action |
|---|---|
| 1 | Use the project browser to select the **General Information** sub-directory of the **Documentation** directory. |
| 2 | Select **Open** in the contextual menu.<br>**Result**: The **General Information** window appears.<br>  <br>Details of window *(see page 1829)* |

| Step | Action |
|---|---|
| 3 | Right click in the entry field, positioning the mouse on the hyperlink to be deleted<br>**Result**: The contextual menu appears.<br>  |
| 4 | Click on **Delete Hyperlink** in the contextual menu.<br>**Result**: The **Delete Hyperlink** window appears.<br>  |
| 5 | You do not have access to the **Text to display** or **Confirm hyperlink deletion** entry fields. |
| 6 | Click **OK** to confirm your choice or click **Cancel** to cancel. |

# Section A.9
## Hyperlinks and drag-and-drop

## Drag/drop

### Principle

You can use the drag/drop function to create a hyperlink. This function can be used to:

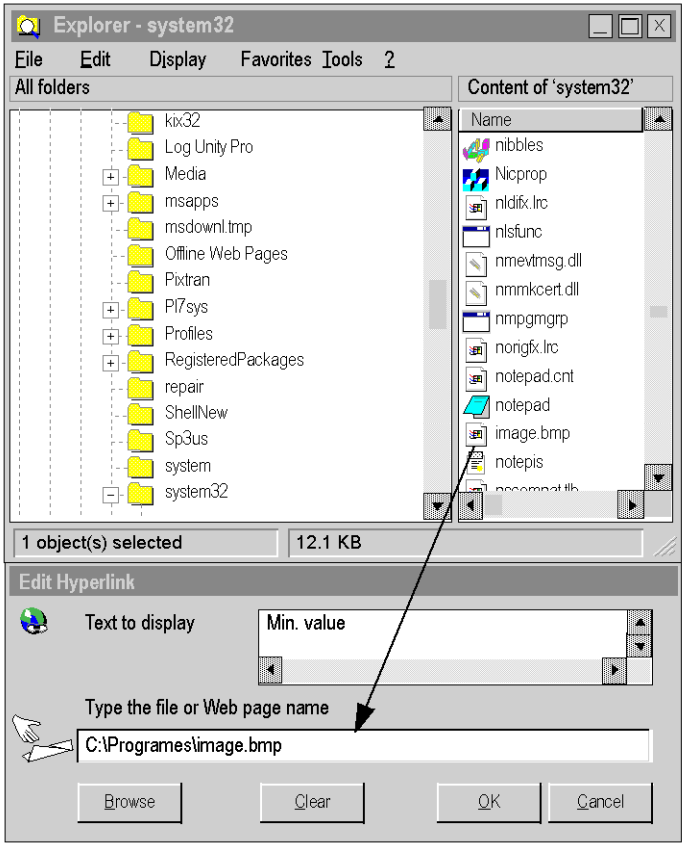
- drag and drop the path name of the target document between the explorer and the **Hyperlink Properties**, **Edit Hyperlink** or **Insert Hyperlink** dialog boxes,
- drag and drop a word or expression between a text document and the comment in the **General Information** window.

**NOTE:** only paths for **.bmp** fillies can be used for drag and drop.

### Procedure

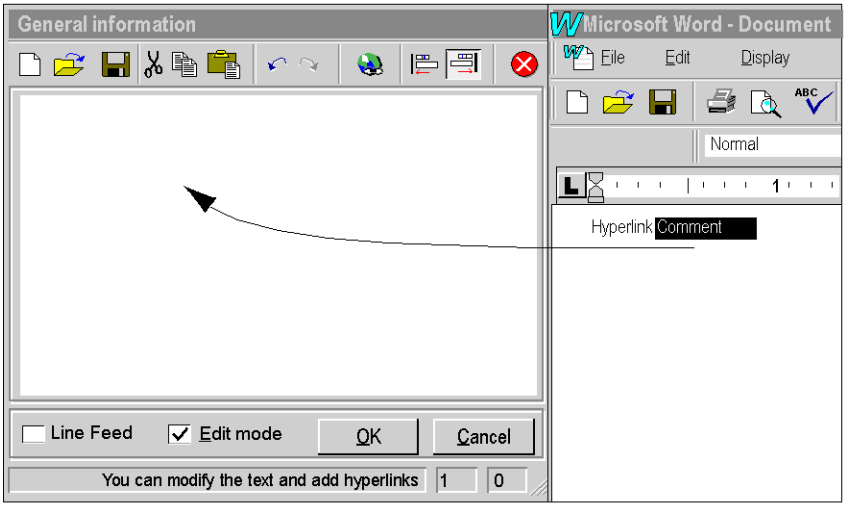
The table below shows the procedure for dragging/dropping between the explorer and the **Hyperlink Properties**, **Insert Hyperlink** or **Edit Hyperlink** dialog boxes.

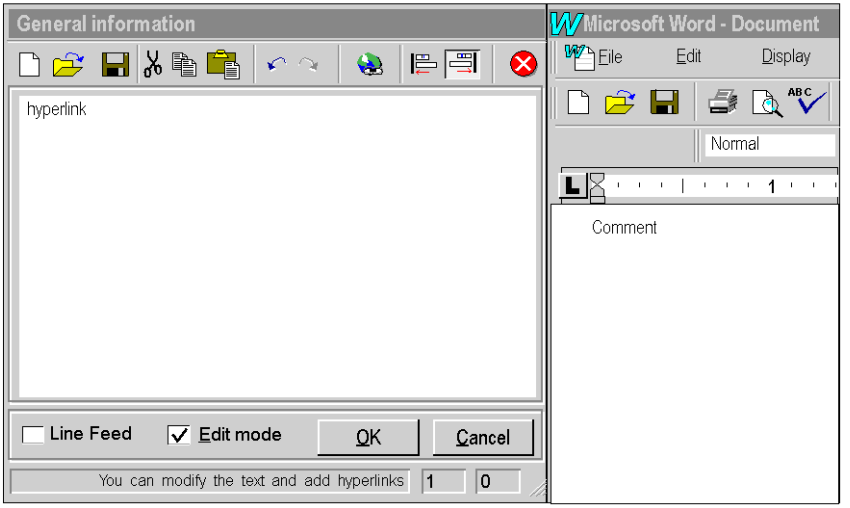
| Step | Action |
|---|---|
| 1 | Locate the target document in the explorer. |
| 2 | In Unity Pro, open either the **Hyperlink Properties**, **Edit Hyperlink** or **Insert Hyperlink** dialog box. |
| 3 | Size and move the Unity Pro and explorer windows so they are displayed side by side on the screen without overlapping. |

| Step | Action |
|---|---|
| 4 | Drag and drop the file representing the target document between the explorer and the **Type the file or Web page name** field of the dialog box (image.bmp in our example).<br>**Result**: The entire path of the target document appears in the **Type the file or Web page name** field.<br>  |

## Procedure

The table below shows the procedure for dragging and dropping between a text file and the **General Information** dialog box.

| Step | Action |
|---|---|
| 1 | In Unity Pro, open the **General Information** window from the **Documentation** directory. |
| 2 | Open the text file. |
| 3 | Size the Unity Pro and explorer windows to display them side by side on the screen. |
| 4 | Select the text to be moved in the text file.   |

| Step | Action |
|---|---|
| 5 | Drag and drop a word or expression between the **General Information** window and a word-processing application.<br>**Result:** The text selected appears in the Hyperlink field.<br>  <br>**Note**: If you press and hold down the Ctrl key, a "+" sign will appear on the mouse cursor. When you drop the expression or word ("Hyperlink" in this case), it will be copied and not moved from the word processor to the comment editor. |

# Appendix B
# User Interface

## Overview

This chapter describes the Unity Pro user interface. It provides an overview of the elements used and how they are used.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Window | 1886 |
| Anchored window | 1889 |
| Menus | 1892 |
| Menu commands | 1895 |
| Toolbar | 1898 |
| Output windows | 1903 |
| Status bar | 1905 |
| Directory Structures | 1907 |
| Using the Mouse | 1909 |
| Using the Keyboard | 1910 |
| Data selection boxes | 1911 |
| Dialog boxes | 1913 |

# Window

## Introduction

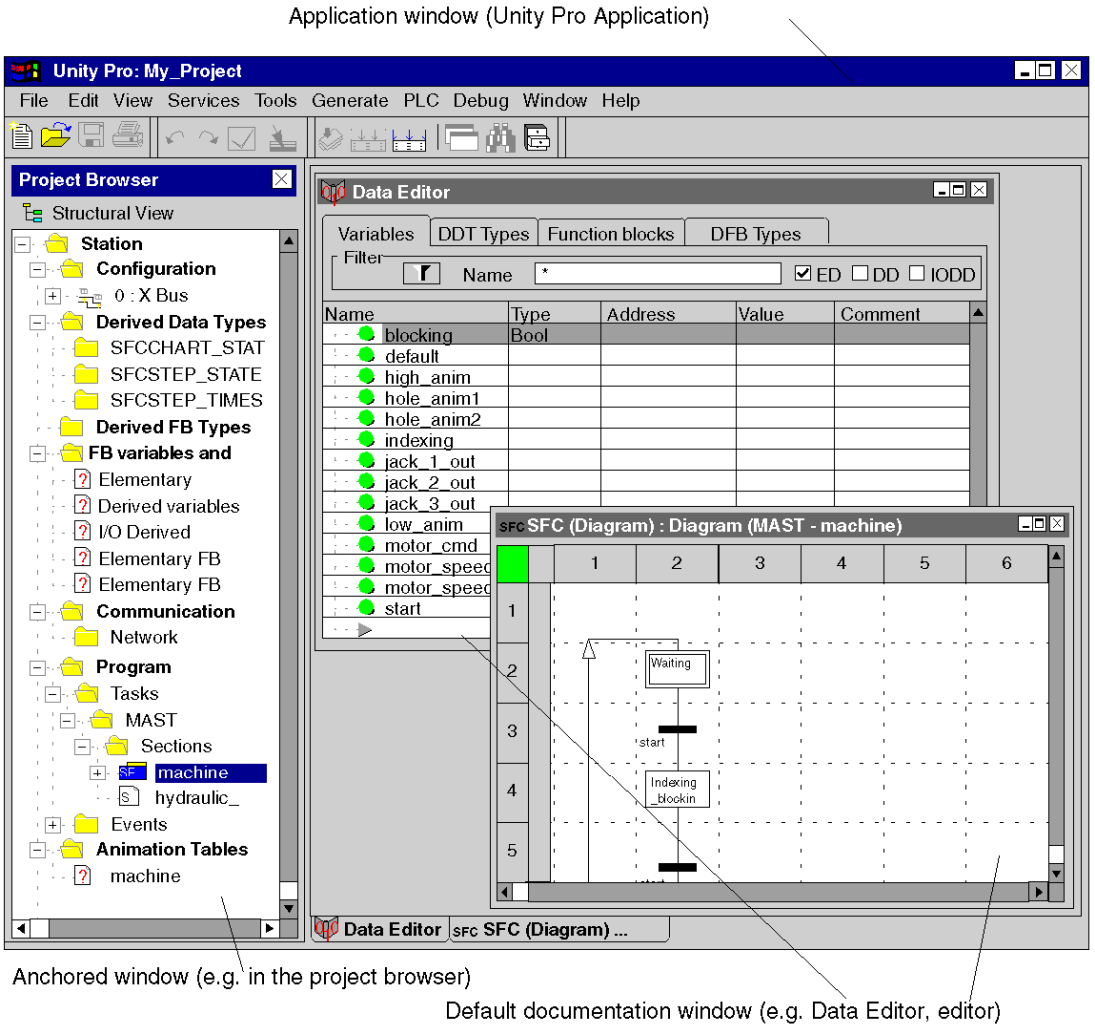
In Windows there are two types of windows:

- Application windows
- Documentation windows

Windows consist of:

- a title bar to move the window and to change its size
- Buttons to display the window as a symbol, to minimize, maximize and close.
- the actual contents of the window

Types of windows:




## Application windows

When you start Unity Pro, an application window will open on your desktop. You can place the application window at any location on your desktop. Or you can minimize it so that it appears in your taskbar.

In this application window you can open or create a project. In the title bar of the application window the name of the project appears.

## Documentation windows

When you open or create a project you can open various documentation windows. For example, documentation windows are sections, where you create your application or the documentation window for the Data Editor.

The information contained in documentation windows is separated into different areas. The different areas in a dialog box can be selected using the **F6** key.

If the amount of information contained in a documentation window is too complex to be displayed in one documentation window, the documentation window is split into several tabs. The different tabs can be selected using the **Ctrl**+**Tab** key combination.

Several documentation windows can be open at the same time, but only one documentation window can be active. An active documentation window is marked by the color of the title bar. Depending on the documentation window the menu commands in the pulldown menu *(see page 1892)* and the toolbar *(see page 1898)* change.

If several windows are open at the same time you can switch between them using the key combination **Ctrl**+**F6**.

An exception to this are anchored windows *(see page 1889)*

# Anchored window

## Introduction

Anchored windows have basically the same function as standard windows with the difference that they can be anchored in certain positions. Anchored windows can also be freely positioned on the screen (also outside of the application window).

## Build

Anchored windows consist of:

- a title bar to move the window and to change its size
- a button to prevent anchoring (  )
- a button to close the window ( **x** )
- the actual contents of the window

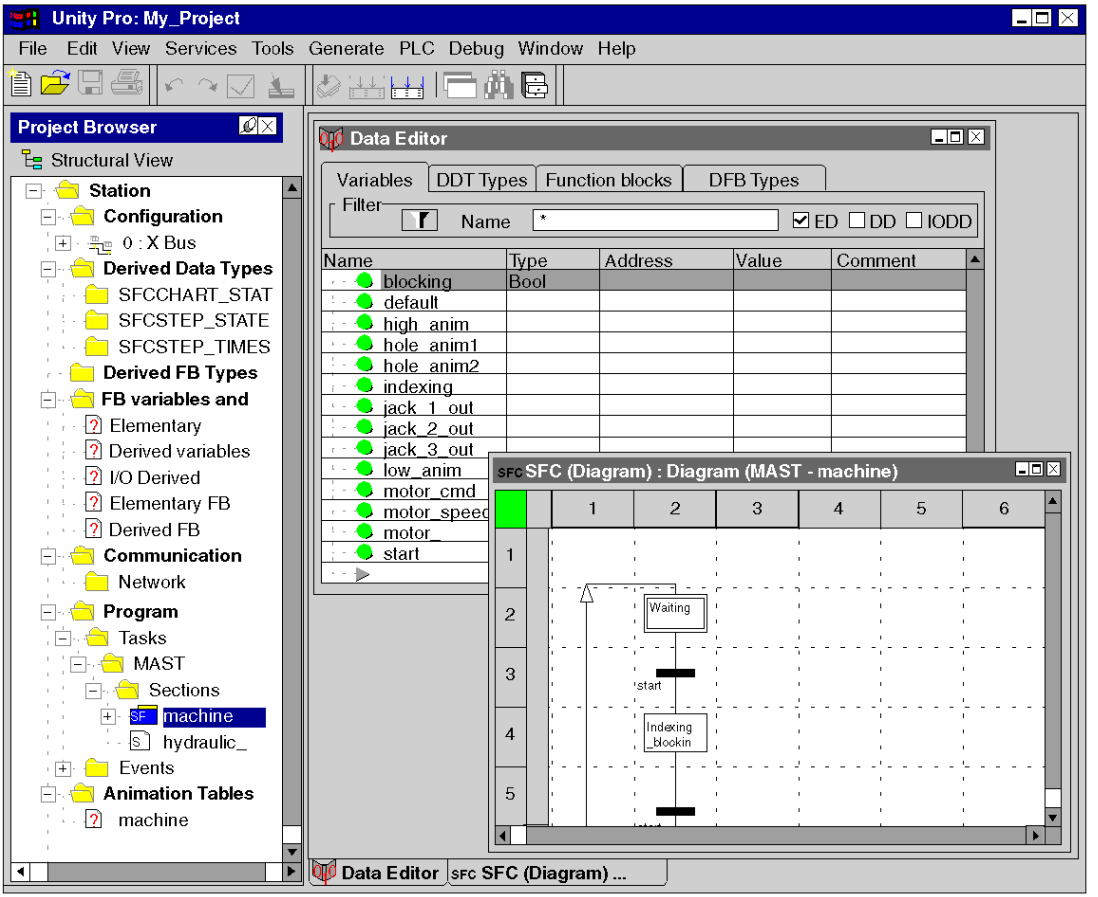
## Function

Anchored windows have the following extra functions in comparison to standard windows:

- They can be anchored to the edge of the application window.
- They can be positioned outside of the application window.
- The menu commands in the **Window** (like e.g. **Arrange**) do not affect the anchored window.

## Example of use

An example of an anchored window is the project browser:



## Anchoring windows

Process the following steps to anchor a window:

| Step | Action |
|---|---|
| 1 | Click the title bar of the anchored window and keep the mouse button pressed. |
| 2 | Drag the window close to an application window edge.<br>**Result:** The window frame displays the future position of the window. |
| 3 | Release the mouse button.<br>**Result:** The window is anchored to the selected edge.<br>On the new position, the window is<br>● anchored<br>when the new position of the window is on the border or outside of the application window.<br>● not anchored<br>when the new position of the window is inside the application window<br>or<br>when the new position of the window is on the border or outside of the application window, but the **Ctrl** key was held when releasing the mouse button. |

# Menus

## Introduction

There are 3 types of menus:

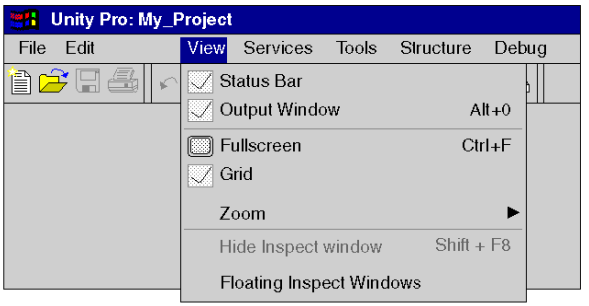
1. Main or pulldown menus
2. Submenus
3. Shortcut or popup menus

## Main or pulldown menus

The titles of the individual menus are displayed in the menu bar. The individual menu commands *(see page 1895)* are listed in the pulldown menus.

A pulldown menu is opened by left-clicking on the title of the menu or by pressing **Alt+selected letter** (underlined letter *(see page 1895)*). 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 or by pressing **Esc**.

Menu bar with pulldown menu:

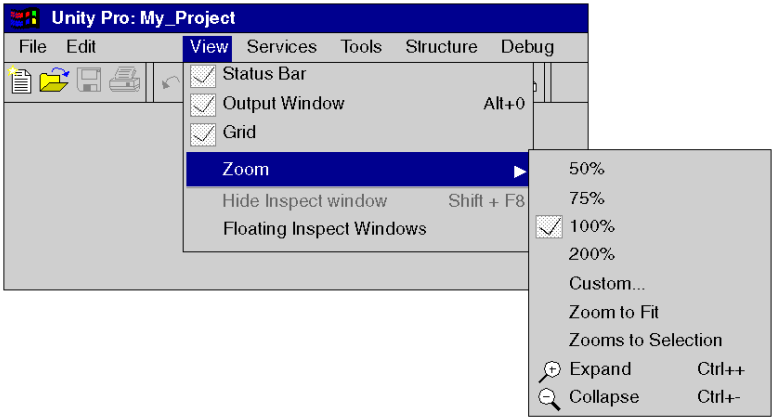



## Submenus

The title of a submenu is a menu command *(see page 1895)* of the menu above it. The individual menu commands for the submenu are listed in a pulldown menu. Menu commands which contain a submenu can be recognized by an arrow icon.

You can open a submenu by clicking on the menu title (left mouse button) or by placing the mouse pointer on the menu title for a short time. The menu can be closed by clicking on the title of the menu or anywhere outside of the menu or by pressing **Esc**.

Menu with submenu:



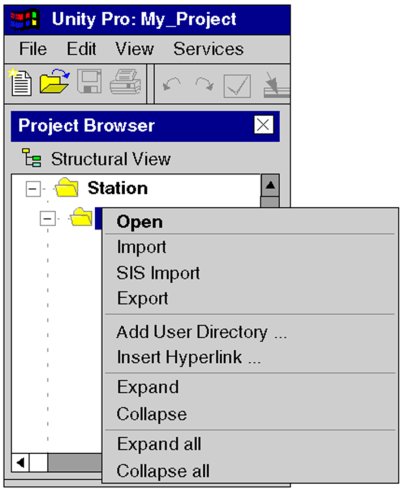

## Shortcut or popup menus

Shortcut menus are menus which contain menu commands *(see page 1895)* specially for the selected object.

You can open a shortcut menu by clicking on the object (right mouse button) or selecting the object and confirming with **Shift** + **F10** or press the key. The menu can be closed by clicking anywhere outside of the menu or by pressing **Esc**.

Shortcut menus can also be called if several objects are selected. In this case, the menu only contains the menu commands which are valid for all objects.

Object with shortcut menu:

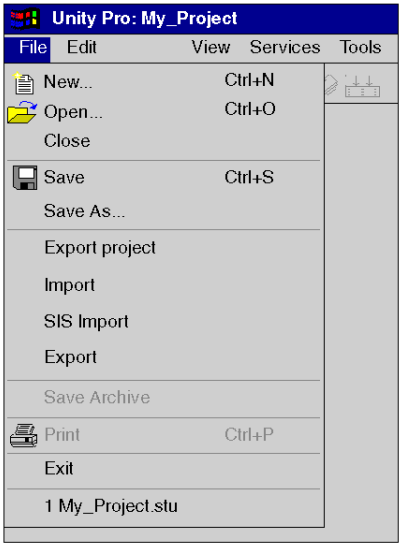


# Menu commands

## Introduction

Menu commands are used to execute commands or to call dialog boxes.

Example of a menu with menu commands:




## Keyboard shortcuts or mnemonics

Keyboard shortcuts (underlined letters) in menu commands allow you to select menu commands using the keyboard. 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.

For example, in the **File** you want to use the menu command **Save...** you must press **Alt**+**D** to open the menu and then **Alt**+**S** to execute the menu command.

## Grayed out menu command

If a menu command is not available it is grayed out. One or more other commands must be executed before the desired menu command can be executed.

## Periods () after the menu command

On execution of this menu command a dialog box is opened with options, which must be selected before execution.

### Checkmark (√) in front of the menu command

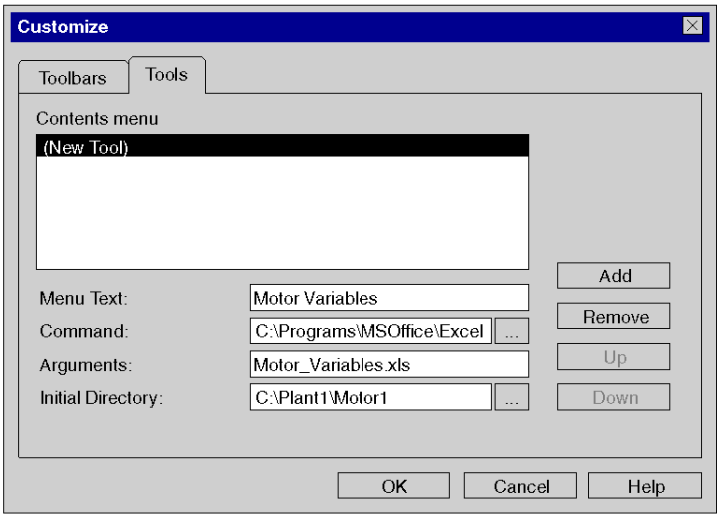
The menu command is enabled. If the menu command is selected, the checkmark disappears and the menu command is disabled. The checkmark is mostly used to identify enabled modes (e.g. normal display, dial in mode etc.).

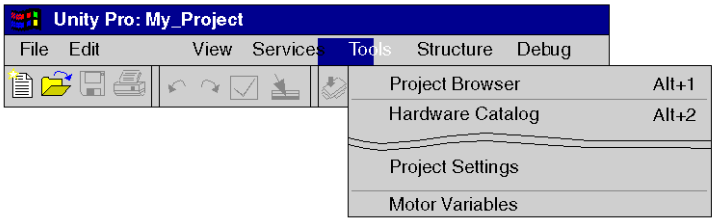
### Shortcut keys

Shortcut keys (e.g. **F8**) or key combinations (e.g. **Ctrl**+**R**) after the menu command is a shortcut way for executing the menu command. You can select the menu command using this shortcut key or key combination without having to open the menu. For example, **Ctrl**+**S** to execute the menu command **Save**.

### Starting external applications

Process the following steps to launch external applications from Unity Pro:

| Step | Action |
|---|---|
| 1 | Open *(see page 571)* **Tools** → **Adjust** the **Tools** tab. |
| 2 | Click the **Add** button. |
| 3 | In the **Menu Text** field, enter the name of the menu item to be created (e.g. `Motor Variables`). |
| 4 | In the **Command** field, enter the path to the program to be launched (e.g. Microsoft Excel). |
| 5 | In the **Arguments** field, enter the name of the file to be opened (e.g. Motor_Variables.xls). |
| 6 | In the **Output Directory** field, enter the path to the file to be opened.<br>Display of the **Tools** tab:<br>  |

| Step | Action |
|---|---|
| 7 | Confirm the entries with **OK**.<br>**Result:** In the main menu **Tools** a menu command to launch the selected application will be generated (in this example **Motor Variables**).<br>**Example:**<br>  |

# Toolbar

## Introduction

Toolbars allow you to quickly locate and execute frequently used functions.

Instead of opening a menu and then selecting the menu command, toolbars allow you to simply click the respective symbol to execute the desired command.

The software provides you with various standard toolbars. In addition, you can also create your own toolbar.

## Build

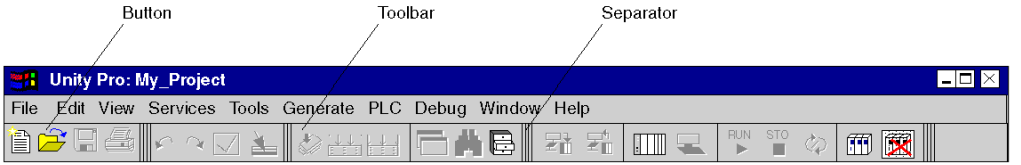
Toolbars consist of a row of buttons and combo boxes that call the corresponding functions.

There are several toolbars and they are displayed in a common toolbar.

The individual toolbars are separated using a separator.

For all items in a toolbar a tooltip can be displayed.

Example of a common toolbar:




## Available toolbars

Standard toolbars are divided into two groups:

- Main menu toolbars
  These toolbars contain the general, i.e. non-context sensitive symbols.
  All available main menu toolbars are displayed when starting the software.
- Context sensitive toolbars
  These toolbars contain the context sensitive symbols.
  The context sensitive toolbars are automatically shown in the active editor.

Depending on the screen resolution it is possible that not all toolbars are visible in the application window.

In this case you have the option to hide the non-required toolbars, or to reposition the toolbars.

## Main menu toolbars

The following main menu toolbars are available by default:

| Name | Contained functions |
|---|---|
| **File** | **New project**, **Open**, **Save**, **Print** |
| **Edit** | **Copy**, **Delete**, **Paste**, **Undo**, **Redo**, **Validate**, **Minimize**, **Maximize**, **Full screen**, **Go to** |
| **Services** | **Analyse project**, **Generate project**, **Re generate entire project**, **Project browser**, **Start search**, **Type library manager** |
| **PLC** | **Transfer Project to PLC**, **Transfer Project from PLC**, **Connect**, **Disconnect**, **Start**, **Stop**, **Start/Stop Animation**, **Standard mode**, **Simulation mode** |
| **Window** | **Cascade**, **Tile Horizontally**, **Tile Vertically** |
| **Help** | **Help**, **What's This** |
| **Debug** | **Set Breakpoint**, **Clear Breakpoint**, **Start**, **Step Over**, **Step Into**, **Step Out**, **Show Current Step**, **Display call stack** |
| **Watchpoint** | **Set Watchpoint**, **Clear Watchpoint**, **Show Watchpoint**, **Synchronize Animation Table**, Refresh counter<br>**Note:** This toolbar is only displayed when a programming language editor is active. |
| **Project Browser** | **Structural view**, **Functional view**, **Vertical views**, **Horizontal views**, **Zoom Out** |

## Showing and hiding

You have three options to hide a toolbar (to close it):

1. Right-click in the common toolbar and deselect the 'undesired' toolbar from the shortcut menu.
2. Open the **Customize** dialog box using **Tools** → **Customize** and deselect the 'undesired' toolbar.
3. For non-anchored toolbars with the close symbol ( **x** ).

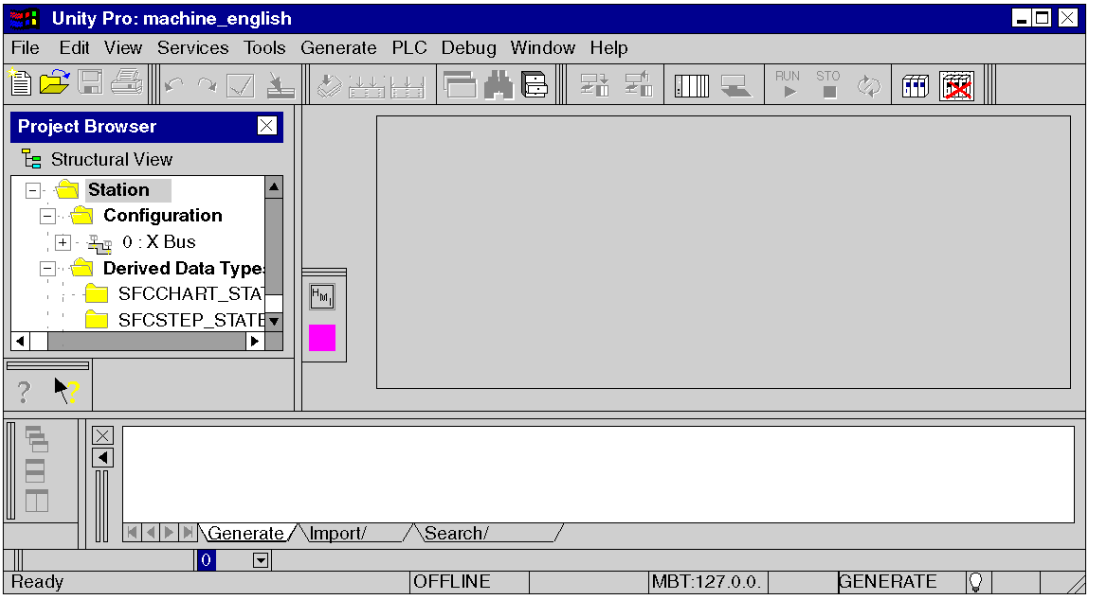
You have two options to show a toolbar (to open it):

1. Right-click in the common toolbar and select the 'desired' toolbar from the shortcut menu.
2. Open the **Customize** dialog box using **Tools** → **Customize** and select the 'desired' toolbar.

## Possible positions

Toolbars can be anchored in the common toolbar (standard) or in different positions inside the application window. Toolbars can also be freely positioned on the screen (also outside of the application window).

Example of different toolbar positions:






## Changing the position of a toolbar

Process the following steps to change the position of a toolbar:

| Step | Action |
|---|---|
| 1 | Click the toolbar separator and hold the mouse button down. |
| 2 | Drag the toolbar to the target position inside or outside the common toolbar.<br>**Result:** The window frame displays the intended position of the toolbar. |
| 3 | Release the mouse button.<br>**Result:** The toolbar is moved to the target position.<br>On the new position, the toolbar is<br>● anchored *(see page 1889)*,<br>when the new position of the toolbar is on the border or outside of the application window<br>● not anchored *(see page 1889)*,<br>when the new position of the toolbar is inside the application window<br>or<br>when the new position of the toolbar is on the border or outside of the application window, but the **Ctrl** key was held when releasing the mouse button |

## Modifying the contents of a toolbar

Process the following steps to modify the contents of a toolbar:

| Step | Action |
|---|---|
| 1 | Open the **Customize** dialog box, see also *Call the Dialog Box:, page 571*. |
| 2 | Click the symbol to be moved in the source toolbar and hold the mouse button down. |
| 3 | Drag the symbol to the target position within the destination toolbar.<br>**Result:** A frame displays the intended position in the toolbar. |
| 4 | Release the mouse button.<br>**Result:** The symbol is removed from the source toolbar and inserted into the destination toolbar. |
| 5 | Click **OK** to confirm the modification(s).<br>**Note:** If symbols are moved from context sensitive toolbars into the main menu toolbar, the symbols are inactive as long as the respective editor is inactive. |

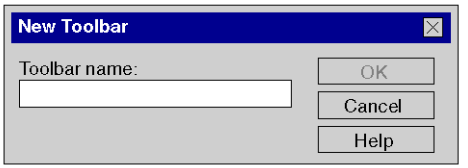
## Restoring toolbars

Process the following steps to restore the original state of a toolbar:

| Step | Action |
|---|---|
| 1 | Open the **Customize** dialog box, see also *Call the Dialog Box:, page 571*. |
| 2 | In the **Toolbars** list box, select the toolbar to be restored. |
| 3 | Click the **Reset** command button.<br>**Result:** The original state of the toolbar is restored. |

## Creating your own toolbars

Process the following steps to create your own toolbars:

| Step | Action |
|---|---|
| 1 | **Note:** New symbols cannot be created. Only existing symbols can be removed from toolbars and inserted into new toolbars.<br>Open the **Customize** dialog box, see also *Call the Dialog Box:, page 571*. |
| 2 | Click **New...**.<br>**Result:** A dialog for defining the toolbar name is opened.<br>  |
| 3 | Enter a name and confirm with **OK**.<br>**Result:** An empty toolbar is created and the name of the toolbar appears in the toolbar list. |
| 4 | Click the symbol to be moved in the source toolbar and hold the mouse button down. |
| 5 | Drag the symbol to the target position within the new toolbar.<br>**Result:** A frame displays the intended position in the toolbar. |
| 6 | Release the mouse button.<br>**Result:** The symbol is removed from the source toolbar and inserted into the destination toolbar. |
| 7 | Click **OK** to confirm the modification(s).<br>**Note:** If symbols are moved from context sensitive toolbars into the main menu toolbar, the symbols are inactive as long as the respective editor is inactive. |
| 8 | Drag the new toolbar to the target position, see also *Changing the position of a toolbar, page 1901*. |

# Output windows

## Introduction

The output window displays information about the different processes (generate, import/export, user error, search/replace).

The output window is an anchored window *(see page 1889)*.

The output window is displayed at the bottom of the Unity Pro window by default.

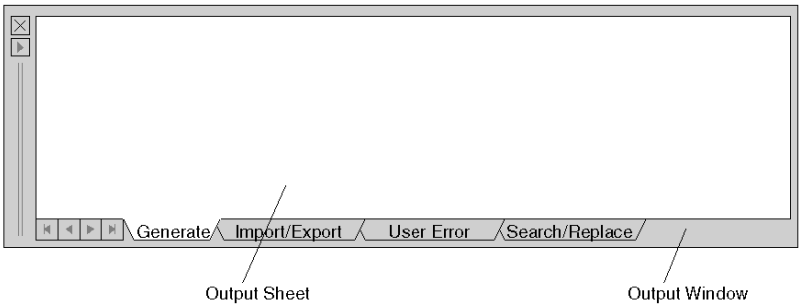
If the output window is closed, it can be opened using:

- the **View** → **Output Window** menu command
  or
- the **Alt+0** key combination.

## Build

The output window consists of various output sheets. Each sheet corresponds to a tab.

Representation of an output window:




The output sheets display the results of an executed action or error messages.

Double-clicking an error message will open the respective editor and mark the "faulty" entry.

New error entries are displayed in red, already checked errors are displayed in blue.

Available output sheets:

| Output sheet | Function |
|---|---|
| Analyze and generate | Displays analyze and generate errors.<br>Double-clicking an error entry directly accesses the faulty programming language section, the faulty configuration or the faulty data. |
| Import/Export | Displays import and export errors<br>Double-clicking an error entry directly accesses the error in the source file. |
| User error | Displays the user error. |

| Output sheet | Function |
|---|---|
| Search/Replace | Displays the result of the search and/or replace operation. Double-clicking an entry directly accesses the searched for entry in the programming language section, in the configuration or in the data. |
| Comparison | Displays the comparison information. |

Every output sheet has a shortcut menu (right-click).

### Copy

Copies the selected text to the clipboard.

### Delete

Deletes the entire contents of the output sheet.

### Go to previous error

Jumps to the previous line and calls the faulty programming language section, the faulty configuration or the faulty data.

This function can also be accessed pressing the **Ctrl**+**F10** key combination.

### Go to next error

Jumps to the next line and calls the faulty programming language section, the faulty configuration or the faulty data.

This function can also be accessed pressing the **F10** key.

### Go to error

Calls the faulty programming language section, the faulty configuration or the faulty data as indicated in the current line.

### Print

Prints the contents of the output sheet.

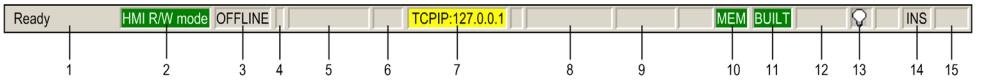
# Status bar

## Introduction

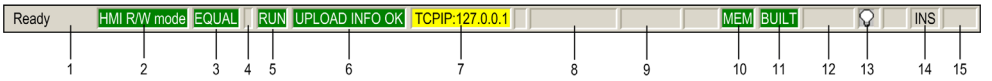
The status bar displays information about the current project on the PC, about the PLC and about the software status.

## Status Bar Description with Examples

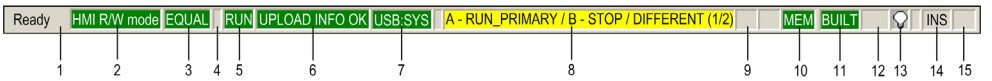
Offline status bar:




Online status bar:




Online Hot standby status bar:




1. Information area for displaying menu information
2. Displays the current HMI access rights
   An application can have different modes:
   - **HMI R/W mode** - Human Machine Interface Read/Write mode:
     This is the default mode when opening an application.
   - **HMI R/O mode** - Human Machine Interface Read only mode:
     This mode is active, when:
     Applications are opened in Read Only mode, by checking the option "Open the project in read-only mode" in the open file window.
     Opening an application previously opened in Write mode with third party software (using Pserver). A pop up window alerts the user at the opening.
     This mode does not allow any savings of the application.
     When Using third party application with Pserver, Unity Pro XL can be launch without the Unity Pro graphical Interface generally called as a Human Machine Interface. In this case, a dialog box prompt the user if it should switch on the current PServer Instance.
3. Link status (**OFFLINE**, **DIFFERENT**, **EQUAL**)
4. An **"*"** is displayed when the initial value in the project is different/modified from the initial value on the PLC
   To remove **"*"** it is advisable to **Update Local Init Values with PLC Init Values** (Or **Transfer Project from PLC** and save it if parameters has been modified in adjust screens of specific modules …)
5. PLC status (**RUN**, **STOP**, etc.)

**6.** ETS Information (Empty Terminal Support)
**7.** Address of the connected PLC
**8.** Hot Standby status and number of changes (only available for M580)
**9.** Line and column information (only available in the programming language editors)
**10.** Memory status:
- Green: **Function Pack for Memory** is not needed
- Red: **Function Pack for Memory** *(see page 152)*command is recommended

**11.** Generate status (**BUILT**, **NOT BUILT**)
**12.** In the connected equal mode the red ” F“ indicates that the bits or words have been forced in the PLC. Click on this “F” to display animation table containing an exhaustive list of the variables (bits/ words) forced
**13.** Event information
If a PLC detects an error, click the button for details about the detected error.
**14.** Indicates whether insert or overwrite mode is active.
**15.** Indicates whether the Caps lock button is active.

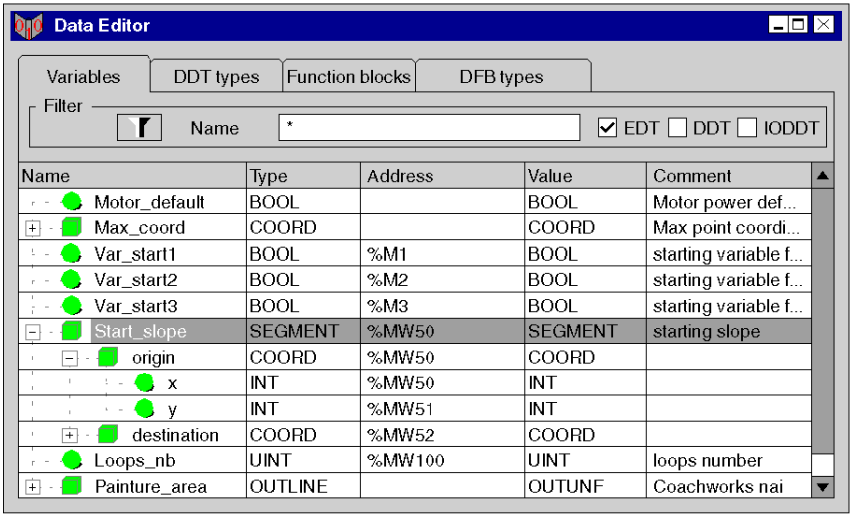
# Directory Structures

## Introduction

Directory structures are used to display and edit hierarchical lists with several columns. Directory structures are combinations of directory trees and lists.

## Example of use

In the Data Editor, directory structures are used to display the hierarchy between variable instances, variable types, structure elements, etc.




## Meaning of the text colors

The text colors have the following meaning:

| Text color | Meaning |
|---|---|
| black | Information is correct and can be edited |
| gray | Information is correct and write protected |
| red | Information is incorrect |

## Modes

There are two different modes for directory structures:

- **Select mode**
  One or more lines are selected or a cell is selected.
  In this mode, you can move between all lines and cells.
- **Edit mode**
  The current cell is opened for editing.
  Texts can be edited.

Switching from the selection mode to the edit mode is done by pressing the **Enter** key or by entering a character, see the **Open editing session by holding down the character key** check box in the *Data and Languages, page 559* dialog box.

Switching from edit mode to selection mode is done by pressing the **Enter** key or by pressing an error key.

# Using the Mouse

## Mouse Functions

The following mouse functions are available:

| Mouse function | In select mode | In edit mode |
|---|---|---|
| Click on links | • Click +/- button:<br>Open/close the substructure.<br>• Click on cell:<br>Selects the lines in the cell clicked and the cell becomes the current cell.<br>• Click on check box:<br>Activate/deactivate the check box<br>• Click on icon:<br>- | • Click +/- button:<br>Saves the information entered (if valid) and opens/closes the substructure.<br>• Click on cell:<br>Saves the information entered (if valid) and selects the lines in the cell clicked.<br>• Click on check box:<br>Saves the information entered (if valid) and activates/deactivates the check box<br>• Click on icon:<br>Saves the information entered (if valid). |
| Double-click on links | • Double-click on names:<br>Selects the lines in the cell clicked and opens the cell in edit mode.<br>• Double-click on the vertical line in the column header:<br>Column width is adjusted to the current contents.<br>• Double-click on the column header cell:<br>Lists are sorted according to column contents | • Double-click on names:<br>Saves the information entered (if valid) and selects the lines in the cell clicked und opens the cell in edit mode.<br>• Double-click on the vertical line in the column header:<br>Saves the information entered (if valid) and the column width is adjusted to the current contents.<br>• Double-click on the column header cell:<br>Saves the information entered (if valid) and lists will be sorted according to the column contents. |
| SHIFT+Click on links | Selects consecutive elements.<br>The first and last element selected and all elements in between are marked. | Saves the information entered (if valid) and selects the lines in the cell clicked. |
| Right click | Opens the shortcut menu (if available) for the object currently being clicked. | Saves the information entered (if valid) and selects the lines in the cell clicked. |

# Using the Keyboard

## Key functions

The following key functions are available:

| Key function | In select mode | In edit mode |
|---|---|---|
| **Enter** | Opens the current cell in edit mode | Saves the information entered (if valid). |
| **Esc** | - | Ends edit mode without saving. |
| **Up arrow** | Deselects the current cell and selects the previous cell (upward). | Saves the information entered (if valid), deselects the current cells and selects the previous cell (upwards). |
| **Cursor down** | Deselects the current cell and selects the next cell (downwards). | Saves the information entered (if valid), deselects the current cells and selects the previous cell (downwards). |
| **Left arrow** | Deselects the current cell and selects the previous cell (left). | Saves the information entered (if valid), deselects the current cells and selects the previous cell (to the left). |
| **Right arrow** | Deselects the current cell and selects the next cell (right). | Saves the information entered (if valid), deselects the current cells and selects the next cell (to the right). |
| **Shift+Up arrow** | Selects multiple lines in upward direction. | - |
| **Shift+Down arrow** | Selects multiple lines in downward direction. | - |
| **+** | Open the substructure. | - |
| **-** | Close the substructure. | - |
| **Character** | Goes to the next lines in which the contents begin with that character. Exceptions: **+**, **-** and **space**. | - |
| **F2** | Opens the current cell in edit mode | - |
| **Insert** | Adds an empty line before the selected line. | - |
| **Home** | The first cell in the selected line becomes the current cell. | Sets the insertion point at the beginning of the current line. |
| **Ctrl+Home** | Selects the first line in the table and the first cell becomes the current cell. | Sets the insertion point at the beginning of the first line. |
| **End** | The last cell in the selected line becomes the current cell. | Sets the insertion point at the end of the current line. |
| **Ctrl+End** | Selects the last line in the table and the first cell becomes the current cell. | Sets the insertion point at the end of the last line. |

# Data selection boxes

## Introduction

Data selection boxes are used to select compatible, already defined data from a list or to select from a list of previous entries.

In this case, "data" means instances and types of variables and instances and types of functions and function blocks.

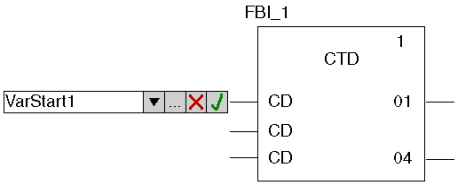
## Build

Data selection boxes consist of:

- a combined text selection and text entry field
- a button to open the selection list (▼)
- a button to open the data selection dialog box ( **...** )
- a button to close the data selection dialog box and delete the (✕) data
- a button to close the data selection dialog box and transfer the (✓) data

## Examples of use

In the language editors, data selection boxes are used to select variables with the correct data type.




## Data entry

There are three options for data entry:

1. Enter the name if known or enter the first letter(s) in the text box.
   The system works with a context dependent background list which automatically completes the characters entered with the first compatible name.
2. Selecting the name in the list box.
   When entering the first letters of the name being searched in the text box, the list box receives an alphabetic list of names that correspond to this entry.
   Without entering the first letters, the list box contains a list of previous entries.
3. Open the data selection dialog box using the ( **...** ).

### Accepting data

With the **Enter** key or the ✓ button, the entry is acknowledged and accepted after having been confirmed successfully.

### Cancel

The entry can be canceled using the **Esc** key or the ✕ button.

# Dialog boxes

## Introduction

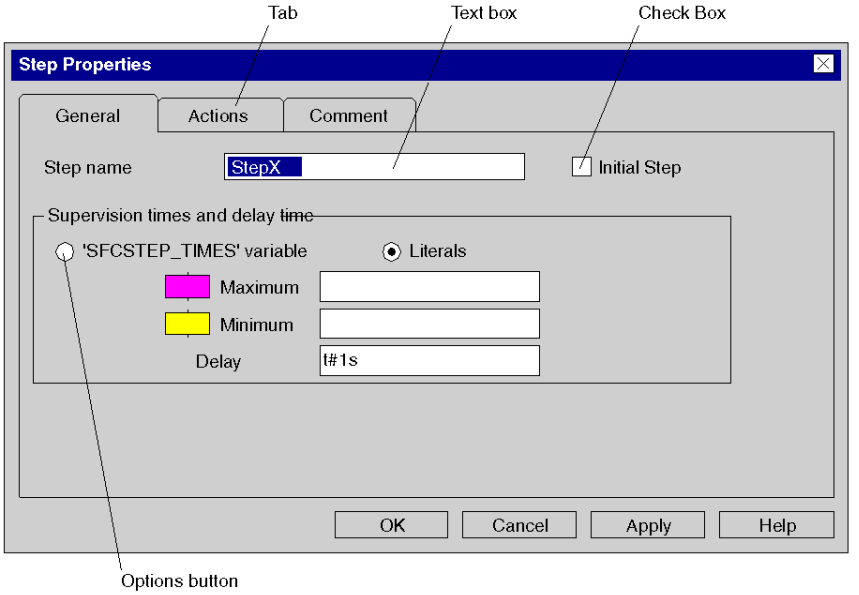
Dialog boxes are displayed if additional information is required from you in order to perform a particular task. Information that you may require is also communicated in this way.

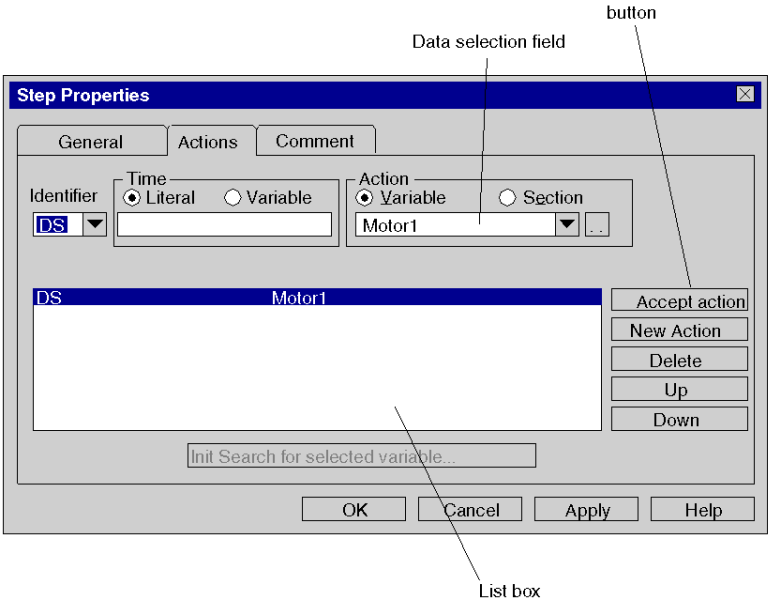
Most dialog boxes contain options which can be selected, text boxes in which text can be entered, and buttons which can be pressed.

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.

The information contained in dialog boxes is separated into different areas. The different area in a dialog box can be selected using the **F6** key.

Basic elements of a window:






## Tab

If the amount of information contained in a dialog box is too complex to be displayed in one dialog box, the dialog box is split into several tabs.

The different tabs in a dialog box can be selected using the **Ctrl**+**Tab** key combination.

## button

buttons are used to initiate actions immediately, e.g. executing or aborting a command. buttons are e.g. **OK**, **Cancel** and **Apply**.

buttons followed by three dots (**...**), open another dialog box. A button with a "greater than" sign (**>>**) extends the active dialog box.

The default setting is identified by a darker border. This default button can also be selected by pressing **Enter**.

If the name of the button contains an underlined letter, the button can be activated from any position in the dialog box by holding down the **Alt** key and entering the underlined letter.

To close a dialog box without executing a command, select the **Cancel** button.

### Text box

Information (text) is entered in a text box.

If you go to an empty text box the cursor appears in the far left of the box. The entered text begins at the cursor position. If text is already present within the respective text box, the text will be selected and replaced by the new text automatically. The text can, however, also be deleted by pressing the **Del** or **backspace** key.

If the name of the text box contains an underlined letter, the text box can be activated from any position in the dialog box by holding down the **Alt** key and entering the underlined letter.

### List box

The available options to select are given in list form in a list box. If more options are available than fit into the list, then the scrollbar or arrow keys can be used to move around the list.

Normally only a single entry can be chosen form the list. There are, however, some cases in which several entries can be chosen, e.g. when opening sections.

If the name of the list box contains an underlined letter, the list box can be activated from any position in the dialog box by holding down the **Alt** key and entering the underlined letter.

### Single line list boxes

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 options to select opens. If more options are available than fit into the list, then the scrollbar or arrow keys can be used to move around the list.

If the name of the list box contains an underlined letter, the list box can be activated from any position in the dialog box by holding down the **Alt** key and entering the underlined letter.

### Data selection boxes

See section *Data selection boxes, page 1911*

### Directory structures

See section *Directory Structures, page 1907*

### 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 the **Alt** key and entering the underlined letter.

## Check boxes

A checkbox next to an option means that the option can be activated or deactivated. Any number of checkbox options can be activated.

Activated options are identified by an **X** or a check mark ( √ ).

If the name of the option contains an underlined letter, the checkbox can be activated from any position in the dialog box by holding down the **Alt** key and entering the underlined letter.

# Appendix C
# Keyboard Shortcuts

## Overview

This chapter describes the keyboard shortcuts of the Unity Pro program.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| C.1 | General Keyboard Shortcuts | 1918 |
| C.2 | Configuring the Project | 1922 |
| C.3 | Creating the Project | 1923 |
| C.4 | Operating the Project | 1935 |
| C.5 | Documenting the Project | 1941 |

# Section C.1
# General Keyboard Shortcuts

## Object of this section

This section provides general information about commonly used keyboard shortcuts. It also provides information about a widely-used browser, namely the Project Browser.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Bus Editor Keyboard Shortcuts | 1919 |
| Unity Pro Main Screen and Project Browser Keyboard Shortcuts | 1920 |

# Bus Editor Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Bus Editor are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the Bus Editor keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Open the New Device windows | Edit | Ctrl+Enter |
| Open the PLC Configuration windows | Edit | Ctrl+Alt+Enter |

# Unity Pro Main Screen and Project Browser Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Unity Pro main screen and in the Project Browser when a project is opened are described in the table below.

## Keyboard Shortcuts

This table describes the Unity Pro main screen and the Project Browser keyboard shortcuts when a project is opened:

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Create a project | File | Ctrl+N |
| Open an existing project | File | Ctrl+O |
| Save an existing project | File | Ctrl+S |
| Print part of an existing project | File | Ctrl+P |
| Display properties | Edit | Alt+Enter |
| Display a full screen | View | Ctrl+F8 |
| Undo the preceeding action | Edit | Ctrl+Z |
| Redo the preceeding action | Edit | Ctrl+Y |
| Cut an item | Edit | Ctrl+X |
| Copy an item | Edit | Ctrl+C |
| Select all an item | Edit | Ctrl+A |
| Paste an item | Edit | Ctrl+V |
| Delete an item | Edit | Del |
| Show or hide the ouput windows | View | Alt+0 |
| Animate the project | Services | Ctrl+J |
| Initialize Search | Services | Ctrl+U |
| Create an animation table | Services | Ctrl+T |
| Display the Project Browser | Tools | Alt+1 |
| Display the Hardware Catalog | Tools | Alt+2 |
| Display the Types Library Browser | Tools | Alt+3 |
| Display the Operator Screen Library | Tools | Alt+4 |
| Search for Cross-References | Tools | Alt+5 |
| Display the Diagnostic Viewer | Tools | Alt+6 |
| Display the PLC Screen | Tools | Alt+7 |
| Display the Variable Window | Tools | Alt+8 |

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Open the Data Editor | Tools | Alt+9 |
| Insert | Edit | Ins |
| Display the Data Properties | Edit | Ctrl+Enter |
| Analyze a Type | Build | Ctrl+Shift+B |
| Control the Changes | Build | Ctrl+B |
| Connect or disconnect a PLC | PLC | Ctrl+K |
| Transfer a project to PLC | PLC | Ctrl+L |
| Transfer a project from a PLC | PLC | Ctrl+Shift+L |
| Put a Quantum Safety PLC in maintenance or safety mode | PLC | Ctrl+Shift+M |
| Run or stop a PLC | PLC | Ctrl+R |
| What's this | Help | Shift+F1 |

**NOTE:** When a project is closed, the Tools and PLC tab shortcuts in the PLC menu are not available but it is still possible to use the Ctrl+K shortcut. The shortcuts from the File, View and Help tab shortcuts are available.

**NOTE:** Commands without shortcuts are not mentionned in this table.

# Section C.2
# Configuring the Project

## PLC Configuration Screen Keyboard Shortcuts

### Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the PLC Configuration Screen are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

### Keyboard Shortcuts

This table describes the PLC Configuration Screen keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Validate the action | Edit | Ctrl+W |
| Copy an input from the list | Edit | Ctrl+C |
| Paste an input in the list | Edit | Ctrl+V |
| Delete an AS_I Slave | Edit | Del |
| Cut an AS_I Slave | Edit | Ctrl+X |
| Copy an AS_I Slave | Edit | Ctrl+C |
| Paste an AS_I Slave | Edit | Ctrl+V |
| Delete a Series 7 module | Edit | Del |
| Cut a Series 7 module | Edit | Ctrl+X |
| Copy a Series 7 module | Edit | Ctrl+C |
| Paste a Series 7 module | Edit | Ctrl+V |

**NOTE:** The Delete, cut, copy paste AS_I Slave shortcuts are only available with Premium Communication TSX SAY 100x modules.

**NOTE:** The Delete, cut, copy paste Series 7 modules shortcuts are only available with Premium Communication TSX ESY 007 modules. Furthermore, these shortcuts are available if the user has specific access rights. Otherwise, the access is denied.

# Section C.3
## Creating the Project

### Object of this section

This section provides information about the keyboard shortcuts used to create a project.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| ST/IL Editors Keyboard Shortcuts | 1924 |
| FBD Editor Keyboard Shortcuts | 1926 |
| SFC Editor Keyboard Shortcuts | 1928 |
| LD Editor Keyboard Shortcuts | 1930 |
| Data Editor Keyboard Shortcuts | 1932 |
| Type Library Browser Keyboard Shortcuts | 1933 |
| Type Library Manager Keyboard Shortcuts | 1934 |

# ST/IL Editors Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the ST/IL Editors are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the ST/IL Editors keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Select data from a list | Edit | Ctrl+D |
| Select the Function Input Assistant | Edit | Ctrl+I |
| Create an IF Statement | Edit/New | F4 |
| Create a FOR Statement | Edit/New | F5 |
| Create a WHILE Statement | Edit/New | F6 |
| Create a REPEAT Statement | Edit/New | Shift+F6 |
| Create a CASE Statement | Edit/New | Shift+F4 |
| Create a variable | Edit/New | Shift+Enter |
| Edit the selected text in Upper case | Edit | Shift+Alt+U |
| Edit the selected text in lowercase | Edit | ALt+U |
| Find the selected text | Edit | Ctrl+F |
| Find the next selected occurrence | Edit | F3 |
| Find the previous selected occurrence | Edit | Shift+F3 |
| Replace the selected occurrence | Edit | Ctrl+H |
| Go to part of the selected project | Edit | Ctrl+G |
| Add a bookmark | Edit/Bookmark | Ctrl+F2 |
| Go to next bookmark | Edit/Bookmark | F2 |
| Go to previous bookmark | Edit/Bookmark | Shift+F2 |
| Delete all bookmarks | Edit/Bookmark | Ctrl+Shift+F2 |
| Display the general attributes of the selected item | Edit | Ctrl+Enter |
| Modify the properties of the selected item | Edit | Alt+Enter |
| Add a comment | Edit/New | F8 |
| Open the Inspect window | Edit/New | F9 |

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Zoom in | View/Zoom | • Ctrl+Plus (+)<br>• Ctrl+Rotate the mouse wheel |
| Zoom out | View/Zoom | • Ctrl+Minus (-)<br>• Ctrl+Rotate the mouse wheel |
| Show/Hide the Inspect window | View | Shift+F8 |
| Go to the previous section | View | Ctrl+Alt+Pg Up |
| Go to the next section | View | Ctrl+Alt+Pg Dn |
| Refine an object | Services | Ctrl+Q |
| Set breakpoint | Debug | Ctrl+F11 |
| Clear breakpoint | Debug | Ctrl+Shift+F11 |
| Go to the selected breakpoint | Debug | Ctrl+F5 |
| Step over the next action | Debug | F11 |
| Step into the action | Debug | F12 |
| Step out the action | Debug | Shift+F12 |
| Set watchpoint | Debug | Alt+F11 |
| Clear watchpoint | Debug | Alt+Shift+F11 |

**NOTE:** The IF, FOR, WHILE, REPEAT, CASE statement shortcuts are only available with the ST Editor.

# FBD Editor Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the FBD Editor are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the FBD Editor keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Select the Function Input Assistant | Edit | Ctrl+I |
| Create a variable | Edit | Shift+Enter |
| Select data from a list | Edit | Ctrl+D |
| Link an item | Edit/New | F6 |
| Add a comment | Edit/New | F8 |
| Open the Inspect window | Edit/New | F9 |
| Select Mode | Edit | Esc |
| Add a bookmark | Edit/Bookmark | Ctrl+F2 |
| Go to next bookmark | Edit/Bookmark | F2 |
| Go to previous bookmark | Edit/Bookmark | Shift+F2 |
| Delete all bookmarks | Edit/Bookmark | Ctrl+Shift+F2 |
| Zoom to fit | View/Zoom | Ctrl +*<br>(* = multiplication sign of numeric pad) |
| Zoom in | View/Zoom | • Ctrl+Plus (+)<br>• Ctrl+Rotate the mouse wheel |
| Zoom out | View/Zoom | • Ctrl+Minus (-)<br>• Ctrl+Rotate the mouse wheel |
| Show/Hide the Inspect window | View | Shift+F8 |
| Go to the previous section | View | Ctrl+Alt+Pg Up |
| Go to the next section | View | Ctrl+Alt+Pg Dn |
| Refine an object | Services | Ctrl+Q |
| Set breakpoint | Debug | Ctrl+F11 |
| Clear breakpoint | Debug | Ctrl+Shift+F11 |
| Go to the selected breakpoint | Debug | Ctrl+F5 |

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Step over the next action | Debug | F11 |
| Step into the action | Debug | F12 |
| Step out the action | Debug | Shift+F12 |
| Set watchpoint | Debug | Alt+F11 |
| Clear watchpoint | Debug | Alt+Shift+F11 |
| Properties | Edit | Alt+Enter |
| Data Properties | Edit/New | Ctrl+Enter |
| Data Selection | Edit/New | Ctrl+D |
| FFB Input Assistant | Edit/New | Ctrl+I |

**NOTE:** Most of the Edit tab shortcuts are available by selecting Shift+F10.

# SFC Editor Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the SFC Editor are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the STB Editor keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Add a step | Edit/New | F3 |
| Add a macro step | Edit/New | Ctrl+F3 |
| Add a transition | Edit/New | Shift+F3 |
| Add an alternative branch | Edit/New | F4 |
| Add an alternative joint | Edit/New | Shift+F4 |
| Add a parallel branch | Edit/New | F5 |
| Add a parallel joint | Edit/New | Shift+F5 |
| Add a link | Edit/New | F6 |
| Add a comment | Edit/New | F8 |
| Select Mode | Edit | Esc |
| Add a bookmark | Edit/Bookmark | Ctrl+F2 |
| Go to next bookmark | Edit/Bookmark | F2 |
| Go to previous bookmark | Edit/Bookmark | Shift+F2 |
| Delete all bookmarks | Edit/Bookmark | Ctrl+Shift+F2 |
| Zoom to fit | View/Zoom | Ctrl +*<br>(* = multiplication sign of numeric pad) |
| Zoom in | View/Zoom | • Ctrl+Plus (+)<br>• Ctrl+Rotate the mouse wheel |
| Zoom out | View/Zoom | • Ctrl+Minus (-)<br>• Ctrl+Rotate the mouse wheel |
| Expand/come back to the project display | View | Ctrl+E |
| Go to the previous section | View | Ctrl+Alt+Pg Up |
| Go to the next section | View | Ctrl+Alt+Pg Dn |

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Initialize Animation Table | Services | Ctrl+T |
| Initialize Search | Services | Ctrl+U |
| Data Properties | Edit/New | Ctrl+Enter |
| Properties | Edit | Alt+Enter |

**NOTE:** Most of the Edit table shortcuts are available by selecting Shift+F10.

# LD Editor Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the LD Editor are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the LD Editor keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Add a normally open contact | Edit/New | F3 |
| Add a closed contact | Edit/New | Shift+F3 |
| Add a positive transition-sensing contact | Edit/New | Ctrl+F3 |
| Add a negative transition-sensing contact | Edit/New | Ctrl+Shift+F3 |
| Add a coil | Edit/New | F5 |
| Add a negated coil | Edit/New | Shift+F5 |
| Add a set coil | Edit/New | Alt+F5 |
| Add a reset coil | Edit/New | Shift+Alt+F5 |
| Add a call coil | Edit/New | F4 |
| Add a boolean connection | Edit/New | F7 |
| Add a vertical connection | Edit/New | Shift+F7 |
| Add a boolean link | Edit/New | Alt+F6 |
| Add a link | Edit/New | F6 |
| Add an operate block | Edit/New | Alt+F7 |
| Add a compare block | Edit/New | Ctrl+F7 |
| Add a comment | Edit/New | F8 |
| Select mode | Edit | Esc |
| Add a bookmark | Edit/Bookmark | Ctrl+F2 |
| Go to next bookmark | Edit/Bookmark | F2 |
| Go to previous bookmark | Edit/Bookmark | Shift+F2 |
| Delete all bookmarks | Edit/Bookmark | Ctrl+Shift+F2 |
| Modify the properties of the selected item | Edit | Alt+Enter |
| Show or hide the status bar | View | Alt+0 |

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Zoom to fit | View/Zoom | Ctrl +*<br>(* = multiplication sign of numeric pad) |
| Zoom in | View/Zoom | • Ctrl+Plus (+)<br>• Ctrl+Rotate the mouse wheel |
| Zoom out | View/Zoom | • Ctrl+Minus (-)<br>• Ctrl+Rotate the mouse wheel |
| Show/Hide the Inspect window | View | Shift+F8 |
| Display the mixed mode | View | Ctrl+E |
| Go to the previous section | View | Ctrl+Alt+Pg Up |
| Go to the next section | View | Ctrl+Alt+Pg Dn |
| Set breakpoint | Debug | Ctrl+F11 |
| Clear breakpoint | Debug | Ctrl+Shift+F11 |
| Go to the selected breakpoint | Debug | Ctrl+F5 |
| Step over the next action | Debug | F11 |
| Step into the action | Debug | F12 |
| Step out the action | Debug | Shift+F12 |
| Set watchpoint | Debug | Alt+F11 |
| Clear watchpoint | Debug | Alt+Shift+F11 |
| Inspect Window | Edit | F9 |
| Data Selection | Edit | Ctrl+D |
| FFB Input Assistant | Edit | Ctrl+I |
| Properties | Edit | Alt+Enter |
| Data Properties | Edit/New | Ctrl+Enter |
| Refine | Services | Ctrl+Q |

**NOTE:** Most of the Edit tab shortcuts are available by selecting Shift+F10.

# Data Editor Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Data Editor are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the Data Editor keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Hide the Inspect window | View | Shift+F8 |

**NOTE:** Some shortcuts are not available from the menu bar. Please note the following shortcuts.

- Select F2 to edit data
- Select Ctrl+ Home to access the first cell in row
- Select Ctrl+ End to access the last cell in row
- Select + on a node to expand it
- Select - on a node to contract it

# Type Library Browser Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Type Library Browser are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the Type Library Browser keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Refine an object | Services | Ctrl+Q |
| Initialize search | Services | Ctrl+U |

# Type Library Manager Keyboard Shortcuts

## Introduction

Some actions or commands can be performed with the keyboard. The shortcuts that you can use in the Type Library Manager are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the Type Library Manager keyboard shortcuts.

| Action | Keyboard Shortcut |
|---|---|
| Access the first cell in row | Home |
| Access the last cell in row | End |
| Access the next cell up | Up |
| Access the next cell down | Down |
| Access next cell left | Left |
| Access next cell right | Right |
| Expand a node | + |
| Contract a node | - |
| Exit the project | Alt+F4/Esc |
| Delete data | Delete |
| Selected Several EFB | Shift+Left |

# Section C.4
## Operating the Project

### Object of this section

This section provides information about the keyboard shortcuts used to operate a project.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Search Replace Tool Screen Keyboard Shortcuts | 1936 |
| Diagnostic Viewer Screen Keyboard Shortcuts | 1937 |
| PLC Screen Keyboard Shortcuts | 1938 |
| Animation Screen Keyboard Shortcuts | 1939 |
| Operator Screen Keyboard Shortcuts | 1940 |

# Search Replace Tool Screen Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Search Replace Tool Screen are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the Search Replace Tool Screen keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Find Next item | Edit | F3 |
| Find Previous item | Edit | Shift+F3 |
| Replace an item | Edit | Ctrl+H |

**NOTE:** Most of the Edit tab shortcuts are available by selecting Shift+F10.

# Diagnostic Viewer Screen Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. All the shortcuts that you can use in the Diagnostic Viewer Screen are described in the General Keyboard Shortcut General Keyboard shortcuts *(see page 1920)* map.

# PLC Screen Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. All the the shortcuts that you can use in the PLC Screen are described in theGeneral Keyboard Shortcuts General Keyboard shortcuts *(see page 1920)* map.

# Animation Screen Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Animation Screen are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the Animation Screen keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Display format in decimal | View/Display Format | F3 |
| Display format in hexadecimal | View/Display Format | Ctrl+F3 |
| Display format in binary | View/Display Format | Shift+F3 |
| Display format in ASCII | View/Display Format | Alt+F3 |
| Enable Variable Modification | Services | F7 |
| Enable Variable Forcing | Services | F6 |
| Set value to 0 | Services | Shift+F5 |
| Set value to 1 | Services | F5 |
| Force to 0 | Services | Shift+F4 |
| Force to 1 | Services | F4 |
| Unforce | Services | Shift+F6 |
| Multiple Mode | Services | Ctrl+F7 |
| Execute Multiple Values | Services | Alt+F7 |
| Reset Multiple Values | Services | Shift+F7 |
| Safety/Maintenance | PLC | Ctrl+Shift+M |

**NOTE:** Most of the Edit tab shortcuts are available by selecting Shift+F10.

**NOTE:** This shortcut "Ctrl+Shift+M" is only available with XLS

# Operator Screen Keyboard Shortcuts

## Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Operator Screen are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

## Keyboard Shortcuts

This table describes the Operator Screen keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Validate | Edit | Ctrl +W |
| Select all | Edit | Ctrl+A |
| Enable variation modification | Services | F7 |
| Go to the previous animated object | Services | F4 |
| Go to the next animated object | Services | F5 |

**NOTE:** Some shortcuts are not available from the menu bar. Please note the following shortcuts.

- Select Space to select next item
- Select Shift+ Space to select the previous item
- Select Alt+ Backspace to cancel thelast action
- Select Tab to focus on the next control object in online mode
- Select Shift+ Tab to focus on the previous control object in online mode
- Select Enter to display the properties of a selected object
- Select an Arrow to make a one pixel move in the arrow direction
- Select Shift+ Arrow to make a 10 pixel move in the arrow direction
- Select Page Up/Down to move the vertical scrollbar up or down
- Select Ctrl + Left/Right to move the horizontal scrollbar left or right

# Section C.5
# Documenting the Project

## Documentation Screen Keyboard Shortcuts

### Introduction

Numerous actions or commands can be performed with the mouse as well as the keyboard. The shortcuts that you can use in the Documentation Screen are described in the table below. Furthermore, most of the general keyboard shortcuts are available in the General Keyboard shortcuts *(see page 1920)* map.

### Keyboard Shortcuts

This table describes the Documentation Screen keyboard shortcuts.

| Action | Menu | Keyboard Shortcut |
|---|---|---|
| Validate the action | Edit | **Ctrl**+**W** |

**NOTE:** Some shortcuts are not available from the menu bar. Please note the following shortcuts.

- Select F5 to refresh a project
- Select **Ctrl**+**Plus (+)** to zoom in a project
- Select **Ctrl**+**Plus (-)** to zoom out of a project
- Select **Ctrl**+**E** to zoom to fit

# Glossary

## !

**%I**

According to the IEC standard, %I indicates a discrete input-type language object.

**%ID**

According to the IEC standard, %ID indicates an input double word-type language object.

Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

**%IF**

According to the IEC standard, %IF indicates an input real-type language object.

Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

**%IW**

According to the IEC standard, %IW indicates an analog input -type language object.

**%KD**

According to the IEC standard, %KD indicates a constant double word-type language object.

For Premium/Atrium PLCs double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) should be located by an integer type (%MW<i>, %KW<i>). Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

For Modicon M340 PLCs, double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) are not available.

**%KF**

According to the IEC standard, %KF indicates a constant real-type language object.

For Premium/Atrium PLCs double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) should be located by an integer type (%MW<i>, %KW<i>). Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

For Modicon M340 PLCs, double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) are not available.

**%KW**

According to the IEC standard, %KW indicates a constant word-type language object.

For Premium/Atrium PLCs double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) should be located by an integer type (%MW<i>, %KW<i>). Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

For Modicon M340 PLCs, double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) are not available.

**%M**

According to the IEC standard, %M indicates a memory bit-type language object.

**%MD**

According to the IEC standard, %MD indicates a memory double word-type language object.

For Premium/Atrium PLCs double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) should be located by an integer type (%MW<i>, %KW<i>). Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

For Modicon M340 PLCs, double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) are not available.

**%MF**

According to the IEC standard, %MF indicates a memory real-type language object.

For Premium/Atrium PLCs double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) should be located by an integer type (%MW<i>, %KW<i>). Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

For Modicon M340 PLCs, double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) are not available.

**%MW**

According to the IEC standard, %MW indicates a memory word-type language object.

For Premium/Atrium PLCs double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) should be located by an integer type (%MW<i>, %KW<i>). Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

For Modicon M340 PLCs, double-type instances of located data (%MD<i>, %KD<i>) or floating (%MF<i>, %KF<i>) are not available.

**%Q**

According to the IEC standard, %Q indicates a discrete output-type language object.

**%QD**

According to the IEC standard, %QD indicates an output double word-type language object.

Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

**%QF**

According to the IEC standard, %QF indicates an output real-type language object.

Only I/O objects make it possible to locate type instances (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) by using their topological address (for example %MD0.6.0.11, %MF0.6.0.31).

**%QW**

According to the IEC standard, %QW indicates an analog output-type language object.

## A

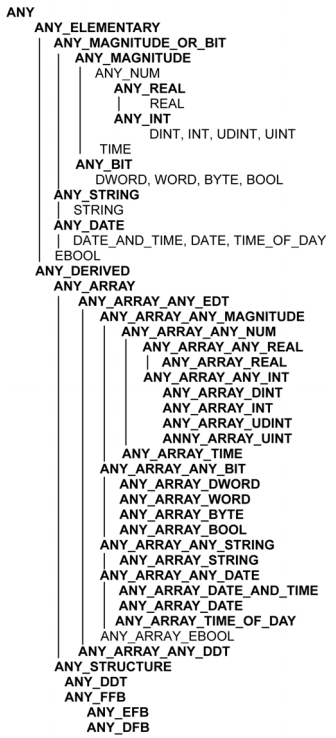
**animating the links**

This is also called **power flow**, and refers to a type of animation used with Ladder language and the function blocks. The links are displayed in red, green or black according to the variables connected.

**ANY**

There is a hierarchy between the different types of data. In the DFB, it is sometimes possible to declare which variables can contain several types of values. Here, we use ANY_xxx types.

The following diagram shows the hierarchically-ordered structure:




## ARRAY

An `ARRAY` is a table of elements of the same type.

The syntax is as follows: `ARRAY [<terminals>] OF <Type>`

Example:

`ARRAY [1..2] OF BOOL` is a one-dimensional table made up of two `BOOL`-type elements.

`ARRAY [1..10, 1..20] OF INT` is a two-dimensional table made up of 10x20 `INT`-type elements.

## ASCII

ASCII is the abbreviation for American Standard Code for Information Interchange.

This is an American code (but which has become an international standard) that uses 7 bits to define every alphanumerical character used in English, punctuation symbols, certain graphic characters and other miscellaneous commands.

**auxiliary tasks**

Optional periodic tasks used to process procedures that do not require fast processing: measurement, adjustment, diagnostic aid, etc.

## B

**base 10 literals**

A literal value in base 10 is used to represent a decimal integer value. This value can be preceded by the signs "+" and "-". If the character "_" is employed in this literal value, it is not significant.

Example:

`-12`, `0`, `123_456`, `+986`

**base 16 literals**

An literal value in base 16 is used to represent an integer in hexadecimal. The base is determined by the number "16" and the sign "#". The signs "+" and "-" are not allowed. For greater clarity when reading, you can use the sign "_" between bits.

Example:

`16#F_F` or `16#FF` (in decimal 255)

`16#F_F` or `16#FF` (in decimal 224)

**base 2 literals**

A literal value in base 2 is used to represent a binary integer. The base is determined by the number "2" and the sign "#". The signs "+" and "-" are not allowed. For greater clarity when reading, you can use the sign "_" between bits.

Example:

`2#1111_1111` or `2#11111111` (in decimal 255)

`2#1110_0000` or `2#11100000` (in decimal 224)

**base 8 literals**

A literal value in base 8 is used to represent an octal integer. The base is determined by the number "8" and the sign "#". The signs "+" and "-" are not allowed. For greater clarity when reading, you can use the sign "_" between bits.

Example:

`8#3_77` or `8#377` (in decimal 255)

`8#34_0` or `8#340` (in decimal 224)

**BCD**

The Binary Coded Decimal (BCD) format is used to represent decimal numbers between 0 and 9 using a group of four bits (half-byte).

In this format, the four bits used to code the decimal numbers have a range of unused combinations.

Example of BCD coding:
- the number 2450
- is coded: 0010 0100 0101 0000

**BIT**

This is a binary unit for a quantity of information which can represent two distinct values (or statuses): 0 or 1.

**BOOL**

BOOL is the abbreviation for Boolean type. This is the elementary data item in computing. A BOOL type variable has a value of either: 0 (FALSE) or 1 (TRUE).

A BOOL type word extract bit, for example: %MW10.4.

**break point**

Used in the "debug" mode of the application.

It is unique (one at a time) and, when reached, signals to the processor to stop the program run.

Used in connected mode, it can be positioned in one of the following program elements:
- LD network,
- Structured Text Sequence or Instruction List,
- Structured Text Line (Line mode).

**BYTE**

When 8 bits are put together, this is callad a BYTE. A BYTE is either entered in binary, or in base 8.

The BYTE type is coded in an 8 bit format, which, in hexadecimal, ranges from 16#00 to 16#FF

## C

**constants**

An INT, DINT or REAL type variable located in the constant field (%K), or variables used in direct addressing (%KW, %KD or %KF). The contents of these cannot be modified by the program during execution.

**CPU**

Is the abbreviation for Control Processing Unit.

This is the microprocessor. It is made up of the control unit combined with the arithmetic unit. The aim of the control unit is to extract the instruction to be executed and the data needed to execute this instruction from the central memory, to establish electrical connections in the arithmetic unit and logic, and to run the processing of this data in this unit. We can sometimes find ROM or RAM memories included in the same chip, or even I/O interfaces or buffers.

**cyclic execution**

The master task is executed either cyclically or periodically. Cyclical execution consists of stringing cycles together one after the other with no waiting time between the cycles.

# D

**DATE**

The DATE type coded in BCD in 32 bit format contains the following information:

- the year coded in a 16-bit field,
- the month coded in an 8-bit field,
- the day coded in an 8-bit field.

The DATE type is entered as follows: **D#**<Year>**-**<Month>**-**<Day>

This table shows the lower/upper limits in each field:

| Field | Limits | Comment |
|---|---|---|
| Year | [1990,2099] | Year |
| Month | [01,12] | The left 0 is always displayed, but can be omitted at the time of entry |
| Day | [01,31] | For the months 01\03\05\07\08\10\12 |
| | [01,30] | For the months 04\06\09\11 |
| | [01,29] | For the month 02 (leap years) |
| | [01,28] | For the month 02 (non leap years) |

**DBCD**

Representation of a Double BCD-format double integer.

The Binary Coded Decimal (BCD) format is used to represent decimal numbers between 0 and 9 using a group of four bits.

In this format, the four bits used to code the decimal numbers have a range of unused combinations.

Example of DBCD coding:

- the number 78993016
- is coded: 0111 1000 1001 1001 0011 0000 0001 0110

**DDT**

DDT is the abbreviation for Derived Data Type.

A derived data type is a set of elements of the same type (ARRAY) or of various types (structure)

**device DDT**

Device DDT is the abbreviation for Device Derived Data Type.

A device derived data type is a predefined DDT that describes the I/O language elements of an I/O Module. This data type is represented in a structure, which depends on the capabilities of the I/O module.

**DFB**

DFB is the abbreviation for Derived Function Block.

DFB types are function blocks that can be programmed by the user ST, IL, LD or FBD.

By using DFB types in an application, it is possible to:
- simplify the design and input of the program,
- increase the legibility of the program,
- facilitate the debugging of the program,
- reduce the volume of the generated code.

**DFB instance**

A DFB type instance occurs when an instance is called from a language editor.

The instance possesses a name, input/output interfaces, the public and private variables are duplicated (one duplication per instance, the code is not duplicated).

A DFB type can have several instances.

**DINT**

`DINT` is the abbreviation for Double Integer format (coded on 32 bits).

The lower and upper limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

`-2147483648`, `2147483647`, `16#FFFFFFFF`.

**documentation**

Contains all the information of the project. The documentation is printed once compiled and used for maintenance purposes.

The information contained in the documentation cover:
- the hardware and software configuration,
- the program,
- the DFB types,
- the variables and animation tables,
- the cross-references.
- ...

When building the documentation file, you can include all or some of these items.

**driver**

A program indicating to your computer's operating system the presence and characteristics of a peripheral device. We also use the term peripheral device driver. The best-known drivers are printer drivers. To make a PLC communicate with a PC, communication drivers need to be installed (Uni-Telway, XIP, Fipway, etc.).

**DT**

DT is the abbreviation for Date and Time.

The DT type coded in BCD in 64 bit format contains the following information:
- The year coded in a 16-bit field,
- the month coded in an 8-bit field,
- the day coded in an 8-bit field,
- the hour coded in a 8-bit field,
- the minutes coded in an 8-bit field,
- the seconds coded in an 8-bit field.

**NOTE:** The 8 least significant bits are unused.

The DT type is entered as follows:

**DT#**<Year>**-**<Month>**-**<Day>**-**<Hour>**:**<Minutes>**:**<Seconds>

This table shows the lower/upper limits in each field:

| Field | Limits | Comment |
|---|---|---|
| Year | [1990,2099] | Year |
| Month | [01,12] | The left 0 is always displayed, but can be omitted at the time of entry |
| Day | [01,31] | For the months 01\03\05\07\08\10\12 |
| | [01,30] | For the months 04\06\09\11 |
| | [01,29] | For the month 02 (leap years) |
| | [01,28] | For the month 02 (non leap years) |
| Hour | [00,23] | The left 0 is always displayed, but can be omitted at the time of entry |
| Minute | [00,59] | The left 0 is always displayed, but can be omitted at the time of entry |
| Second | [00,59] | The left 0 is always displayed, but can be omitted at the time of entry |

**DWORD**

DWORD is the abbreviation for Double Word.

The DWORD type is coded in 32 bit format.

This table shows the lower/upper limits of the bases which can be used:

| Base | Lower limit | Upper limit |
|---|---|---|
| Hexadecimal | 16#0 | 16#FFFFFFFF |
| Octal | 8#0 | 8#37777777777 |
| Binary | 2#0 | 2#11111111111111111111111111111111 |

Representation examples:

| Data Content | Representation in One of the Bases |
|---|---|
| 00000000000010101101110011011110 | 16#ADCDE |
| 00000000000000001000000000000000 | 8#200000 |
| 00000000000010101011110011011110 | 2#10101011110011011110 |

# E

**EBOOL**

`EBOOL` is the abbreviation for Extended Boolean type. A `EBOOL` type variable brings a value (0 (`FALSE`) or 1 (`TRUE`) but also rising or falling edges and forcing capabilities.

An `EBOOL` type variable takes up one byte of memory.

The byte split up into:
- one bit for the value,
- one bit for the history bit (each time the state's object changes, the value is copied inside the history bit),
- one bit for the forcing bit (equals to 0 if the object isn't forced, equal to 1 if the bit is forced.

The default type value of each bit is 0 (`FALSE`).

**EDT**

**EDT** is the abbreviation for Elementary Data Type.

These types are as follows:
- `BOOL`,
- `EBOOL`,
- `WORD`,
- `DWORD`,
- `INT`,
- `DINT`,
- `UINT`,
- `UDINT`,
- `REAL`,
- `DATE`,
- `TOD`,
- `DT`.

**EF**

Is the abbreviation for Elementary Function.

This is a block which is used in a program, and which performs a predefined software function.

A function has no internal status information. Multiple invocations of the same function using the same input parameters always supply the same output values. Details of the graphic form of the function invocation can be found in the "[Functional block (instance)]". In contrast to the invocation of the function blocks, function invocations only have a single unnamed output, whose name is the same as the function. In FBD each invocation is denoted by a unique [number] via the graphic block, this number is automatically generated and can not be altered.

You position and set up these functions in your program in order to carry out your application.

You can also develop other functions using the SDKC development kit.

**EFB**

Is the abbreviation for Elementary Function Block.

This is a block which is used in a program, and which performs a predefined software function.

EFBs have internal statuses and parameters. Even where the inputs are identical, the output values may be different. For example, a counter has an output which indicates that the preselection value has been reached. This output is set to 1 when the current value is equal to the preselection value.

**EN / ENO (enable / error notification)**

EN means **EN**able, this is an optional block input.

If EN = 0, the block is not activated, its internal program is not executed and ENO it is set to 0.

If EN = 1, the internal program of the block is executed, and ENO is set to 1 by the system. If an error occurs, ENO is set to 0.

ENO means **E**rror **NO**tification, this is the output associated to the optional input EN.

If ENO is set to 0 (caused by EN = 0 or in case of an execution error),

- the outputs of function blocks remain in the status they were in for the last correct executed scanning cycle and
- the output(s) of functions and procedures are set to "0".

**NOTE:** If EN is not connected, it is automatically set to 1.

**event processing**

Event processing 1 is a program section launched by an event. The instructions programmed in this section are executed when a software application event (Timer) or a hardware event (application specific module) is received by the processor.

Event processes take priority over other tasks, and are executed the moment the event is detected.

The event process EVT0 is of highest priority. All others have the same level of priority.

**NOTE:** For M340, IO events with the same priority level are stored in a FIFO and are treated in the order in which they are received.
All the timers have the same priority. When several timers end at the same time, the lowest timer number is processed first.
The system word %SW48 counts IO events and telegram processed.

**NOTE:** TELEGRAM is available only for PREMIUM (not on Quantum or M340)

# F

**fast task**

Task launched periodically (setting of the period in the PC configuration) used to carry out a part of the application having a superior level of priority to the Mast task (master).

**FBD**

FBD is the abbreviation for Function Block Diagram.

FBD is a graphic programming language that operates as a logic diagram. In addition to the simple logic blocks (`AND`, `OR`, etc.), each function or function block of the program is represented using this graphic form. For each block, the inputs are located to the left and the outputs to the right. The outputs of the blocks can be linked to the inputs of other blocks to form complex expressions.

**FFB**

Collective term for EF (Elementary Function), EFB (Elementary Function Block) and DFB (Derived Function block)

**flash eprom**

PCMCIA memory card containing the program and constants of the application.

**FNES**

FNES is the abbreviation for Fichiers Neutres d'Entrées Sorties (Neutral I/O Documentation).

FNES format describes using a tree structure the PLCs in terms of rack, cards and channels.

It is based on the CNOMO standard (comité de normalisation des outillages de machines outils).

**FTP**

FTP is the abbreviation for File Transfer Protocol. FTP is the web's file transfer protocol.

**function view**

View making it possible to see the program part of the application through the functional modules created by the user (see Functional module definition).

**functional module**

A functional module is a group of program elements (sections, sub-programs, macro steps, animation tables, runtime screen, etc.) whose purpose is to perform an automation device function.

A functional module may itself be separated into lower-level functional modules, which perform one or more sub-functions of the main function of the automation device.

## G

**gray**

Gray or "reflected binary" code is used to code a numerical value being developed into a chain of binary configurations that can be differentiated by the change in status of one and only one bit.

This code can be used, for example, to avoid the following random event: in pure binary, the change of the value 0111 to 1000 can produce random numbers between 0 and 1000, as the bits do not change value altogether simultaneously.

Equivalence between decimal, BCD and Gray:

| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| BCD | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 |
| Gray | 0000 | 0001 | 0011 | 0010 | 0110 | 0111 | 0101 | 0100 | 1100 | 1101 |

## H

**HTTP**

HTTP is the abbreviation for HyperText Transfer Protocol. HTTP is the protocol for the formatting and transmission of files on the web. HTTP runs on top of TCP/IP (internet) protocols.

**hyperlink**

The hyperlink function enables links to be created between your project and external documents. You can create hyperlinks in all the elements of the project directory, in the variables, in the processing screen objects, etc.

The external documents can be web pages, files (xls, pdf, wav, mp3, jpg, gif, etc.).

# I

**I/O object**

An I/O object is an implicit or explicit language object for an expert function module or a I/O device on a fieldbus. They are of the following types: %Ch, %I, %IW, %ID, %IF, %Q, %QW, % QD, QF, %KW, %KD, %KF, %MW, %MD, and %MF.

The objects' topological address depends on the module's position on the rack or the device's position on the bus.

For Premium/Atrium PLCs double-type instances of located data (`%MD<i>`, `%KD<i>`) or floating (`%MF<i>`, `%KF<i>`) should be located by an integer type (`%MW<i>`, `%KW<i>`). Only I/O objects make it possible to locate type instances (`%MD<i>`, `%KD<i>`, `%QD`, `%ID`, `%MF<i>`, `%KF<i>`, `%QF`, `%IF`) by using their topological address (for example `%MD0.6.0.11`, `%MF0.6.0.31`).

For Modicon M340 PLCs, double-type instances of located data (`%MD<i>`, `%KD<i>`) or floating (`%MF<i>`, `%KF<i>`) are not available.

**IEC 61131-3**

International standard: Programmable Logic Controls

Part 3: Programming languages.

**IL**

IL is the abbreviation for Instruction List.

This language is a series of basic instructions.

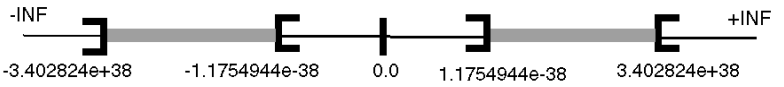
This language is very close to the assembly language used to program processors.

Each instruction is composed of an instruction code and an operand.

**INF**

Used to indicate that a number overruns the allowed limits.

For a number of integers, the value ranges (shown in gray) are as follows:




When a calculation result is:

- less than -3.402824e+38, the symbol `-INF` (for -infinite) is displayed,
- greater than +3.402824e+38, the symbol `+INF` (for +infinite) is displayed.

**instantiate**

To instantiate an object is to allocate a memory space whose size depends on the type of object to be instantiated. When an object is instantiated, it exists and can be manipulated by the program.

**INT**

INT is the abbreviation for single integer format (coded on 16 bits).

The lower and upper limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

-32768, 32767, 2#1111110001001001, 16#9FA4.

**integer literals**

Integer literal are used to enter integer values in the decimal system. The values can have a preceding sign (+/-). Individual underlines (_ ) between numbers are not significant.

Example:

-12, 0, 123_456, +986

**IODDT**

IODDT is the abbreviation for Input/Output Derived Data Type.

The term IODDT designates a structured data type representing a module or a channel of a PLC module. Each application expert module possesses its own IODDTs.

## K

**keyword**

A keyword is a unique combination of characters used as a syntactical programming language element (See annex B definition of the IEC standard 61131-3. All the key words used in Unity Pro and of this standard are listed in annex C of the IEC standard 61131-3. These keywords cannot be used as identifiers in your program (names of variables, sections, DFB types, etc.)).

## L

**LD**

LD is the abbreviation for Ladder Diagram.

LD is a programming language, representing the instructions to be carried out in the form of graphic diagrams very close to a schematic electrical diagram (contacts, coils, etc.).

**located variable**

A located variable is a variable for which it is possible to know its position in the PLC memory. For example, the variable Water_pressure, is associated with %MW102. Water_pressure is said to be localized.

# M

**macro step**

A macro step is the symbolic representation of a unique set of steps and transitions, beginning with an input step and ending with an output step.

A macro step can call another macro step.

**master task**

Main program task.

It is obligatory and is used to carry out sequential processing of the PLC.

**mono task**

An application comprising a single task, and so necessarily the Master task.

**multi task**

Application comprising several tasks (Mast, Fast, Auxiliary, event processing).

The order of priority for the execution of tasks is defined by the operating system of the PLC.

**multiple token**

Operating mode of an SFC. In multitoken mode, the SFC may possess several active steps at the same time.

# N

**naming convention (identifier)**

An identifier is a sequence of letters, numbers and underlines beginning with a letter or underline (e.g., name of a function block type, an instance, a variable or a section). If you select the **Extended** option in the **Tools** → **Project Settings...** → **Variables** dialog, letters from national character sets (e.g., ö, ü, é, õ) can be used. Underlines are significant in identifiers; e.g., `A_BCD` and `AB_CD` are interpreted as different identifiers. Ending underlines is invalid.

Identifiers cannot contain spaces. Not case sensitive; e.g., `ABCD` and `abcd` are interpreted as the same identifier.

According to IEC 61131-3 leading digits are not allowed in identifiers. Nevertheless, you can use them if you activate the check box **Allow leading digits** in the **Tools** → **Project Settings...** → **Variables** dialog.

According to IEC 61131-3 multiple leading underlines and consecutives underlines are not allowed in identifiers. Nevertheless, you can use them if you select the **Extended** option in the **Tools** → **Project Settings...** → **Variables** → **Character set** dialog.

Identifiers cannot be keywords.

**NAN**

Used to indicate that a result of an operation is not a number (NAN = Not A Number).

Example: calculating the square root of a negative number.

**NOTE:** The IEC 559 standard defines two classes of NAN: quiet NAN (QNAN) and signaling NaN (SNaN) QNAN is a NAN with the most significant fraction bit set and a SNAN is a NAN with the most significant fraction bit clear (Bit number 22). QNANs are allowed to propagate through most arithmetic operations without signaling an exception. SNAN generally signal an invalid-operation exception whenever they appear as operands in arithmetic operations (See %SW17 and %S18).

**network**

Mainly used in communication, a network is a group of stations which communicate among one another. The term network is also used to define a group of interconnected graphic elements. This group forms then a part of a program which may be composed of a group of networks.

## O

**operator screen**

This is an editor that is integrated into Unity Pro, which is used to facilitate the operation of an automated process. The user regulates and monitors the operation of the installation, and, in the event of any problems, can act quickly and simply.

## P

**periodic execution**

The master task is executed either cyclically or periodically. In periodic mode, you determine a specific time (period) in which the master task must be executed. If it is executed under this time, a waiting time is generated before the next cycle. If it is executed over this time, a control system indicates the overrun. If the overrun is too high, the PLC is stopped.

**procedure**

Procedures are functions view technically. The only difference to elementary functions is that procedures can take up more than one output and they support data type VAR_IN_OUT. To the eye, procedures are no different than elementary functions.

Procedures are a supplement to IEC 61131-3.

**protection**

Option preventing the contents of a program element to be read (read protected), or to write or modify the contents of a program element (read/write protected).

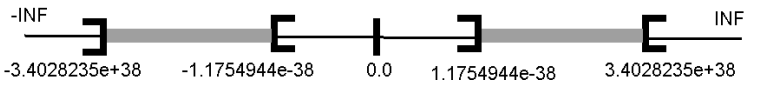
The protection is confirmed by a password.

# R

**REAL**

Real type is a coded type in 32 bits.

The ranges of possible values are illustrated in gray in the following diagram:




When a calculation result is:

- between -1.175494e-38 and 1.175494e-38 it is considered as a `DEN`,
- less than -3.4028234e+38, the symbol `-INF` (for - infinite) is displayed,
- greater than +3.4028234e+38, the symbol `INF` (for +infinite) is displayed,
- undefined (square root of a negative number), the symbol `NAN` or `NAN` is displayed.

**NOTE:** The IEC 559 standard defines two classes of NAN: quiet `NAN` (`QNAN`) and signaling `NaN` (`SNaN`) `QNAN` is a `NAN` with the most significant fraction bit set and a `SNAN` is a `NAN` with the most significant fraction bit clear (Bit number 22). `QNAN`s are allowed to propagate through most arithmetic operations without signaling an exception. `SNAN` generally signal an invalid-operation exception whenever they appear as operands in arithmetic operations (See %SW17 and %S18).

**NOTE:** when an operand is a `DEN` (De-normalized number) the result is not significant.

**real literals**

An literal real value is a number expressed in one or more decimals.

Example:

`-12.0`, `0.0`, `+0.456`, `3.14159_26`

**real literals with exponent**

An Literal decimal value can be expressed using standard scientific notation. The representation is as follows: mantissa + exponential.

Example:

`-1.34E-12` or `-1.34e-12`

`1.0E+6` or `1.0e+6`

`1.234E6` or `1.234e6`

**reference data type**

A reference data type is a data type that allows to create a variable which contains the address of another variable.

This specific data type is described in the Language Reference manual *(see Unity Pro, Program Languages and Structure, Reference Manual )*.

**RS 232C**

Serial communication standard which defines the voltage of the following service:

- a signal of +12 V indicates a logical 0,
- a signal of -12 V indicates a logical 1.

There is, however, in the case of any attenuation of the signal, detection provided up to the limits -3 V and +3 V.

Between these two limits, the signal will be considered as invalid.

RS 232 connections are quite sensitive to interference. The standard specifies not to exceed a distance of 15 m or a maximum of 9600 bauds (bits/s).

**RS 485**

Serial connection standard that operates in 10 V/+5 V differential. It uses two wires for send/receive. Their "3 states" outputs enable them to switch to listen mode when the transmission is terminated.

**run**

Function enabling the startup of the application program of the PLC.

**run auto**

Function enabling the execution of the PLC application program to be started automatically in the case of a cold start.

**rung**

A rung is the equivalent of a sequence in LD; other related terms are "Ladder network" or, more generally, "Network". A rung is inscribed between two potential bars of an LD editor and is composed of a group of graphic elements interconnected by means of horizontal or vertical connections. The dimensions of a rung are 17 to 256 lines and 11 to 64 columns maximum.

## S

**section**

Program module belonging to a task which can be written in the language chosen by the programmer (FBD, LD, ST, IL, or SFC).

A task can be composed of several sections, the order of execution of the sections corresponding to the order in which they are created, and being modifiable.

**SFC**

SFC is the abbreviation for Sequential Function Chart.

SFC enables the operation of a sequential automation device to be represented graphically and in a structured manner. This graphic description of the sequential behavior of an automation device, and the various situations which result from it, is performed using simple graphic symbols.

**SFC objects**

An SFC object is a data structure representing the status properties of an action or transition of a sequential chart.

**single token**

Operating mode of an SFC chart for which only a single step can be active at any one time.

**ST**

ST is the abbreviation for Structured Text language.

Structured Text language is an elaborated language close to computer programming languages. It enables you to structure series of instructions.

**STRING**

A variable of the type `STRING` is an ASCII standard character string. A character string has a maximum length of 65534 characters.

**structure**

View in the project navigator with represents the project structure.

**subroutine**

Program module belonging to a task (MAST, FAST, AUX) which can be written in the language chosen by the programmer (FBD, LD, ST, or IL).

A subroutine may only be called by a section or by another subroutine belonging to the task in which it is declared.

## T

**task**

A group of sections and subroutines, executed cyclically or periodically for the MAST task, or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the PLC. These I/O are refreshed in consequence.

**TIME**

The type `TIME` expresses a duration in milliseconds. Coded in 32 bits, this type makes it possible to obtain periods from 0 to (2 to the power of 32)-1 milliseconds.

**time literals**

The units of type `TIME` are the following: the days (`d`), the hours (`h`), the minutes (`m`), the seconds (`s`) and the milliseconds (`ms`). A literal value of the type `TIME` is represented by a combination of previous types preceded by `T#`, `t#`, `TIME#` or `time#`.

Examples: `T#25h15m`, `t#14.7S`, `TIME#5d10h23m45s3ms`

**time out**

In communication projects, the time out is a delay after which the communication is stopped if there is no answer of the target device.

**TOD**

TOD is the abbreviation for Time of Day.

The TOD type coded in BCD in 32 bit format contains the following information:
- the hour coded in a 8-bit field,
- the minutes coded in an 8-bit field,
- the seconds coded in an 8-bit field.

**NOTE:** The 8 least significant bits are unused.

The Time of Day type is entered as follows: **TOD#**<Hour>**:**<Minutes>**:**<Seconds>

This table shows the lower/upper limits in each field:

| Field | Limits | Comment |
|---|---|---|
| Hour | [00,23] | The left 0 is always displayed, but can be omitted at the time of entry |
| Minute | [00,59] | The left 0 is always displayed, but can be omitted at the time of entry |
| Second | [00,59] | The left 0 is always displayed, but can be omitted at the time of entry |

Example: TOD#23:59:45.

**token**

An active step of an SFC is known as a token.

## U

**UDINT**

UDINT is the abbreviation for Unsigned Double Integer format (coded on 32 bits) unsigned. The lower and upper limits are as follows: 0 to (2 to the power of 32) - 1.

Example:

0, 4294967295, 2#11111111111111111111111111111111, 8#37777777777, 16#FFFFFFFF.

**UINT**

UINT is the abbreviation for Unsigned integer format (coded on 16 bits). The lower and upper limits are as follows: 0 to (2 to the power of 16) - 1.

Example:

0, 65535, 2#1111111111111111, 8#177777, 16#FFFF.

**unlocated variable**

An unlocated variable is a variable for which it is impossible to know its position in the PLC memory. A variable which have no address assigned is said to be unlocated.

# V

**variable**

Memory entity of the type BOOL, WORD, DWORD, etc., whose contents can be modified by the program during execution.

**visualization window**

This window, also called a watch window, displays the variables that cannot be animated in the language editors. Only those variables that are visible at a given time in the editor are displayed.

# W

**watch point**

Used in the "debug" mode of the application.

It enables the display of animated variables to be synchronized with the execution of a program element (containing the watch point) in order to ascertain their values at this precise point of the program.

**WORD**

The WORD type is coded in 16 bit format and is used to carry out processing on bit strings.

This table shows the lower/upper limits of the bases which can be used:

| Base | Lower Limit | Upper Limit |
|---|---|---|
| Hexadecimal | 16#0 | 16#FFFF |
| Octal | 8#0 | 8#177777 |
| Binary | 2#0 | 2#1111111111111111 |

Representation examples

| Data Content | Representation in One of the Bases |
|---|---|
| 0000000011010011 | 16#D3 |
| 1010101010101010 | 8#125252 |
| 0000000011010011 | 2#11010011 |

# Index




## A

adding
  user directory, *177*
adjusting
  variables, *1483*
animation
  animation tables, *1491*
  inspect window, *1488*
  PLC screen, *1470*
  program, *1406*
  variables window, *1485*
animation tables
  hyperlinks, *1860*
application, *190*
  protecting, *180*
assistant
  calling FFB functions, *1194*
  import/export, *1761*
  type library manager, *307*

## B

backup project
  Modicon M340, *132*
  Premium, *134*
bookmarks
  graphical languages, *576*
browser
  DTM, *262*
Browser Contextual Menus, *273*
  Field Bus Discovery, *281*
browser directories, *193*
budget
  IO, *1344*

## C

coils
  LL984 editor, *1111*
comparing
  project, *114*
configuration editor, *1287*
configuring
  project, *1287*
configuring networks, *427*
configuring X-way, *435*
connecting to PLC, *99*
  communication parameters, *111*
  setting address, *101*
  types of connection, *107*
contacts
  LL984 editor, *1111*, *1134*
Container
  FDT, *253*
conversion wizard
  access, *1781*
  conversion, *1786*
  CPU configuration tab, *1813*
  general procedure, *1779*
  general representation, *1784*
  I/O remapping tab, *1792*
  introduction, *1779*
  launch, *1781*
  results, *1786*
  structure tab, *1787*
creating
  derived data types, *333*
  EDT/DDT/IODDT/Device DDT instances, *369*
  EFB/DFB instances, *359*
creating a program
  LL984 editor, *1111*, *1112*
current library, *299*

## D

data editor, *325*
  hyperlinks, *1841*
  management, *403*

data storage, *190*
  protecting, *188*
DDT (derived data type), *333*
  instances, *369*
debugging
  program, *1395*
derived data types (DDT), *333*
  instances, *369*
derived function blocks (DFB), *1251*
DFB
  hyperlinks, *1849*
DFB (derived function block), *1251*
DFB Type, *1252*
diagnostics, *1535*
diagnostics viewer, *1541*
documentation, *1693*
  hyperlinks, *1874*
DTM
  Browser, *263*
DTM hardware catalog, *257*
DTM Hardware Catalog, *257*
DTM properties, *288*
DTM services, *275*

## E

EDT (elementary data type), *369*
EFB (elementary function block)
  instances, *359*
elementary data types (EDT), *369*
elementary function blocks (EFB)
  instances, *359*
equation network blocks
  LL984 editor, *1111*, *1163*
Excel
  export, *1741*
  import, *1744*
export, *1719*
  Excel, *1741*

## F

FBD
  reference data type, *705*
FBD editor, *611*
  creating a program, *612*
  editing FFBs, *630*
  other functions, *575*
  search/replace function, *1235*
  selecting data, *1181*
  using type library browser, *1225*
FDT Container, *253*
FDT container
  introduction, *255*
  user rights, *256*
FFB library version
  properties, *179*
Field Bus Discovery, *281*
filtering
  data, *412*
firmware, *190*
  protecting, *184*
forget
  password, *190*
FTP, *156*
function blocks
  LL984 editor, *1111*, *1152*
functional modules, *221*

## G

go to
  project browser, *174*
graphical languages
  bookmarks, *576*

## H

Hardware Catalog
  DTM, *257*
hyperlinks, *1818*

## I

I/O objects management, *1201*
IL
  reference data type, *1071*

IL editor, *1041*
other functions, *1007*
search/replace function, *1235*
selecting data, *1181*
using type library browser, *1225*
import, *1719*
Excel, *1744*
import XEF file
Libset version, *1757*
import ZEF file
Libset version, *1757*
information
Libset, *296*, *299*
INIT, *131*
instruction editor
LL984 editor, *1522*, *1522*
IO
budget, *1344*
IODDT, *369*

## K

keyboard
shortcut, *1910*

## L

LD
reference data type, *856*
LD editor, *709*
creating a program, *710*
other functions, *575*
search/replace function, *1235*
using type library browser, *1225*
libraries
Libset, *294*
library content, *299*
library management
variables and FFB types, *293*
library manager, *298*
areas, *299*
Libset (set of libraries), *294*
Libset information, *296*, *299*
Libset version, *294*
import XEF file, *1757*
import ZEF file, *1757*
modifying, *295*
options, *85*
properties, *179*
upgrade/downgrade, *295*
links
LL984 editor, *1111*, *1157*
LL984 editor
assigning variables, *1111*, *1171*
coils, *1111*, *1144*
contacts, *1111*, *1134*
creating a program, *1111*, *1112*
equation network blocks, *1111*, *1163*
function blocks, *1111*, *1152*
instruction editor, *1522*, *1522*
links, *1111*, *1157*
network display, *1111*, *1123*
search/replace function, *1235*
subroutines, *1111*, *1175*
loss
password, *190*

## M

maintain output links on disabled EF
project settings, *542*
memory
pack function, *152*
memory card, *156*
memory card access
Modicon M340, *155*
memory consumption, *147*
modifying the Libset version, *295*

## N

network display
LL984 editor, *1111*, *1123*

## O

offline mode, *83*
online mode, *92*

online modifications, *94*
open project dialog, *85*
operator screen, *1571*
  hyperlinks, *1867*
options, *515*
  Libset version, *85*
output windows, *1903*

## P

password
  forget, *190*
  loss, *190*
project browser, *161*
  hyperlinks, *1834*
project settings
  maintain output links on disabled EF, *542*
properties
  FFB library version, *179*
  Libset version, *179*
protecting
  application, *180*
  data storage, *188*
  firmware, *184*
  section, *186*
purge of data, *418*

## R

Ready, *270*
RUN, *131*

## S

search/replace function, *1235*
section
  protecting, *186*
sections, *457*
security editor, *58*
selecting
  data, *1181*
settings
  project, *516*
  toolbar/menu, *571*
  Unity Pro, *554*
SFC editor, *861*
  creating a program, *862*
  other functions, *575*
  search/replace function, *1235*
  selecting data, *1181*
  using type library browser, *1225*
ST
  reference data type, *1108*
ST editor, *1073*
  other functions, *1007*
  search/replace function, *1235*
  selecting data, *1181*
  using type library browser, *1225*
status bar, *1905*
STOP, *131*
structural view, *193*
structure (DDT), *333*
  instances, *369*
subroutines
  LL984 editor, *1111*, *1175*

## T

tasks, *454*
time out
  FTP, *156*
transferring
  data, *136*
  project, *125*
type library
  update, *315*
type library browser, *1225*
type library manager, *293*, *298*
  areas, *299*
  description, *301*

## U

updating
  initial values, *146*
upgrade/downgrade the Libset version, *295*
upload information, *129*

## V

variables (EDT), *369*
variables assignment
    LL984 editor, *1111*, *1171*

## Z

zoom, *582*, *1014*
    project browser, *167*