SHPPLCD
13373835

L-force *Engineering*

Softwarehandbuch

PC-based Automation

PLC Designer

Lenze

Inhaltsverzeichnis

1 Über diese Dokumentation 21

2 Lenze Spezifika 23

2.1 Lenze-Funktionsbausteine und Funktionen 24

2.1.1 Bibliothek: L_PLCLoggingAccess.lib 24

2.1.2 Bibliothek: PLCDataManagerAccess.lib 25

2.1.3 Bibliothek: L_SM3_DriveUtil.lib 27

2.1.4 Bibliothek: L_SM3_DriveUtil.lib (I/O-System 1000) 37

2.2 CANopen / PROFIBUS / EtherCAT 52

2.3 PLC-Programm zur Dokumentation einer Zykluszeit-Verletzung 52

2.4 Auf der SoftMotion-Reiterkarte gemeldete Achsfehler 54

2.5 Task zur Prozessabbild-Übernahme zuordnen 55

2.6 Datenkonsistenz beim Zugriff auf das Prozessabbild 56

3 Konzepte, Basiskomponenten 57

3.1 Einführung und Basiskonzepte 57

3.2 Unterschiede zum PLC Designer V2.3 59

3.3 Profile und Plug-Ins 62

3.4 Projekt 63

3.5 Unterstützte Programmiersprachen 64

3.6 Was ist ein Program Organization Unit Objekt "POU" 64

3.7 Programm-Organisationseinheiten - "POUs" 66

3.7.1 POU 66

3.7.2 Programm 68

3.7.3 Funktion 71

3.7.4 Funktionsblock 73

3.7.5 DUT / Datentyp 82

3.7.6 Methode 84

3.7.7 Eigenschaft 86

3.7.8 Schnittstelle 88

3.7.9 Aktion 90

3.7.10 Externe Funktionen, Funktionsblöcke, Methoden 92

3.7.11 Globale Variablenliste - GVL ... 93
3.7.12 Globale Netzwerkvariablenliste - GNVL ... 93
3.7.13 Persistente Variablen ... 95
3.7.14 Externe Datei ... 96
3.7.15 Textliste ... 98
3.7.16 Bildersammlung ... 104
3.7.17 Visualisierung ... 107
3.7.18 POUs für implizite Prüfungen ... 107
3.8 Gerät, Gerätebaum ... 108
3.9 Applikation ... 114
3.10 Taskkonfiguration ... 116
3.11 Kommunikation ... 117
3.11.1 Steuerungskonfiguration ... 117
3.11.2 Netzwerk Topologie ... 117
3.11.3 Adressierung und Routing ... 118
3.11.4 Adressstrukturen ... 120
3.11.5 Data Server ... 125
3.11.6 Netzwerkvariablen ... 125
3.12 Code-Generierung und Online Change ... 134
3.13 Monitoring ... 137
3.14 Debugging / Fehlersuche ... 137
3.15 Drucken ... 139
3.16 Visualisierung ... 139
3.17 Bibliotheksverwaltung ... 139
3.18 Sicherheit ... 145
4 Quickstart ... 146
4.1 Hilfe ... 146
4.2 Deinstallation, Aktualisieren, Reparieren ... 146
4.3 Projekt erstellen und auf der Steuerung betreiben ... 147

5 Benutzeroberfläche 163

5.1 Komponenten der Benutzeroberfläche 163

5.2 Benutzeroberfläche anpassen 166

5.3 Benutzeroberfläche im Online-Modus 169

6 Standard Menüs und Befehle 171

7 Benutzerverwaltung und Zugriffskontrolle 179

8 Visualisierung 181

8.1 Visualisierung in PLC Designer V3 181

8.2 Voraussetzungen 184

8.3 Optionen 184

8.4 Visualisierungsobjekt anlegen 185

8.5 Visualisierung bearbeiten 185

8.6 Visualization Manager 185

8.7 Startvisualisierung 186

8.8 Frames, Referenzen, Schnittstellen, Platzhalter 186

8.9 Text und Sprache in Visualisierungen 187

8.10 Bilder in einer Visualisierung 190

8.11 Externe Datenquellen 190

8.12 Visualisierungsbibliotheken 191

8.13 Visualisierungsprofile 191

8.14 Target-Visualisierung 192

8.15 Web-Visualisierung 193

8.16 Visualisierung läuft im Programmiersystem 194

8.17 Visualisierung auf externen Geräten - HMI 196

8.18 Visualisierungselemente vom Typ Visualisierung 198

8.19 Visualisierungselemente vom Typ Complex Controls 200

8.19.1 Konfiguration des Elements Meter 201

8.19.2 Konfiguration des Elements Bar Display 204

8.19.3 Konfiguration des Elements Trace 208

8.20 Visualisierungselemente vom Typ Windows Controls 212

8.20.1 Konfiguration des Elements Table 213

8.21 Tastaturbedienung im Online-Modus ... 219
8.22 Events und Eingabeaktionen bemerken ... 220
8.23 Eingabe-Dialoge ... 227

9 Symbolkonfiguration ... 232

10 Quellcodeverwaltung ... 234
10.1 Quellcodeverwaltung über ENI ... 234
10.2 Unterschiede zu PLC Designer V2.3 ... 236

11 Projekt-Dokumentation ... 237

12 Menübefehle, nach Kategorien ... 238
12.1 Befehle und Befehlskategorien ... 238
12.2 Lesezeichen ... 238
12.2.1 Befehle der Kategorie Lesezeichen ... 238
12.2.2 Lesezeichen ein-/ausschalten ... 239
12.2.3 Nächstes Lesezeichen ... 239
12.2.4 Vorheriges Lesezeichen ... 239
12.2.5 Lesezeichen löschen ... 239
12.3 Haltepunkte ... 240
12.3.1 Befehle der Kategorie Haltepunkte ... 240
12.3.2 Haltepunkt umschalten ... 240
12.3.3 Neuer Haltepunkt ... 241
12.3.4 Haltepunkt-Positionen ... 243
12.3.5 Einzelschritt ... 244
12.3.6 Prozedurschritt ... 244
12.3.7 Ausführen bis Rücksprung ... 245
12.3.8 Nächste Anweisung festlegen ... 245
12.3.9 Ausführen bis Cursor ... 245
12.3.10 Nächste Anweisung anzeigen ... 246
12.4 Quelltextnavigation ... 246
12.4.1 Gehe zur Definition ... 246
12.4.2 Querverweisliste ... 247

12.5 Übersetzen 248
12.5.1 Befehle der Kategorie Übersetzen 248
12.5.2 Übersetzen 248
12.5.3 Neu übersetzen 249
12.5.4 Überprüfe alle Pool Objekte 249
12.5.5 Code erzeugen 249
12.5.6 Erzeuge Laufzeitsystemdateien 250
12.5.7 Bereinigen 252
12.5.8 Alles bereinigen 252
12.6 CAM 253
12.6.1 CAM Befehle 253
12.6.2 CAM Online-Datei schreiben 253
12.6.3 CAM-Daten in ASCII-Tabelle schreiben 253
12.6.4 CAM Online-Datei lesen 254
12.6.5 CAM-Daten aus ASCII-Tabelle lesen 254
12.7 CFC 255
12.7.1 CFC Editor Befehle 255
12.7.2 Arbeitsblatt bearbeiten 256
12.7.3 Negieren 258
12.7.4 EN/ENO 258
12.7.5 Set/Reset 259
12.7.6 Ausführungsreihenfolge 260
12.7.7 Parameter bearbeiten 266
12.7.8 Selektierte Anschlüsse verbinden 267
12.7.9 Anschlüsse zurücksetzen 267
12.7.10 Nicht verbundene Anschlüsse entfernen 267
12.8 CNC 268
12.8.1 Befehle der Kategorie CNC 268
12.8.2 Vorverarbeitete Bahn anzeigen 269
12.8.3 Interpolationspunkte anzeigen 269
12.8.4 Satzunterdrückung 270
12.8.5 Raster anzeigen 270

12.8.6 Endpunkte anzeigen ... 270
12.8.7 Bahnansicht scrollen ... 270
12.8.8 Programm verschieben ... 271
12.8.9 Programm skalieren ... 272
12.8.10 Programm drehen ... 273
12.8.11 Richtung umkehren ... 274
12.8.12 CNC-Programm neu nummerieren ... 274
12.8.13 Programm in ASCII-Datei schreiben ... 274
12.8.14 Programm von ASCII-Datei laden ... 274
12.8.15 Von DXF-Format importieren ... 275
12.8.16 Dynamik analysieren ... 276
12.8.17 Variablen setzen ... 278
12.9 Zwischenablage ... 279
12.9.1 Befehle der Kategorie Zwischenablage - Bearbeiten Befehle ... 279
12.9.2 Ausschneiden ... 279
12.9.3 Kopieren ... 280
12.9.4 Einfügen ... 280
12.9.5 Löschen ... 280
12.9.6 Alles auswählen ... 281
12.10 Deklaration ... 281
12.10.1 Tabellarischer Deklarationseditor - Deklaration ... 281
12.10.2 Deklarationskopf editieren ... 281
12.10.3 Einfügen ... 281
12.10.4 Nach unten verschieben ... 281
12.10.5 Nach oben verschieben ... 281
12.11 Geräte ... 282
12.11.1 Geräteobjekte ... 282
12.11.2 Geräte-Scan ... 287
12.11.3 Geräte-Repository ... 293
12.12 Gerätekommunikation ... 295
12.12.1 Befehle der Kategorie Gerätekommunikation ... 295
12.12.2 Gerät hinzufügen ... 295

12.12.3 Gateway hinzufügen... 297
12.12.4 Gerätenamen ändern... 298
12.12.5 Zum lokalen Gateway verbinden... 298
12.12.6 Ausgewähltes Gerät löschen... 305
12.12.7 Gateway bearbeiten... 305
12.12.8 Suche nach Gerät über Name 305
12.12.9 Suche nach Gerät über Adresse 305
12.12.10 Netzwerk durchsuchen... 306
12.12.11 Aktiven Pfad setzen 306
12.13 AWL/FUP/KOP 307
12.13.1 FUP/KOP/AWL Editor Befehle 307
12.13.2 Netzwerk einfügen 307
12.13.3 Netzwerk einfügen unterhalb 308
12.13.4 Kommentierung ein/aus 308
12.13.5 Zuweisung einfügen 309
12.13.6 Bausteinaufruf einfügen 310
12.13.7 Leeren Baustein einfügen 313
12.13.8 Sprung einfügen 314
12.13.9 Sprungmarke einfügen 316
12.13.10 Return einfügen 316
12.13.11 Bausteineingang einfügen 316
12.13.12 Spule einfügen 317
12.13.13 Set-Spule einfügen 317
12.13.14 Reset-Spule einfügen 317
12.13.15 Kontakt einfügen 318
12.13.16 Negierten Kontakt einfügen 319
12.13.17 Kontakt einfügen rechts 319
12.13.18 Kontakt parallel einfügen unterhalb 319
12.13.19 Negierten Kontakt parallel einfügen (unterhalb) 319
12.13.20 Kontakt parallel einfügen oberhalb 320
12.13.21 Kontakte einfügen: Darunter einfügen 320
12.13.22 Kontakte einfügen: Rechts einfügen danach 320

12.13.23 Kontakte einfügen: Darüber einfügen ... 321
12.13.24 AWL-Zeile danach einfügen ... 321
12.13.25 AWL-Zeile löschen ... 321
12.13.26 Negation ... 321
12.13.27 Flankenerkennung ... 322
12.13.28 Set/Reset ... 323
12.13.29 Weiterverschaltung festlegen ... 323
12.13.30 Leitungsverzweigung einfügen ... 324
12.13.31 Leitungsverzweigung oberhalb einfügen ... 325
12.13.32 Leitungsverzweigung unterhalb einfügen ... 325
12.13.33 Parameter aktualisieren ... 325
12.13.34 Nicht verwendete FB-Aufruf-Parameter entfernen ... 326
12.13.35 Ansicht: Als Funktionsbausteinsprache anzeigen ... 326
12.13.36 Ansicht: Als Kontaktplan anzeigen ... 327
12.13.37 Ansicht: Als Anweisungsliste darstellen ... 327
12.14 Suchen/Ersetzen ... 328
12.14.1 Suchen/Ersetzen Befehle ... 328
12.14.2 Suchen ... 329
12.14.3 Ersetzen ... 332
12.14.4 Weitersuchen ... 333
12.14.5 Weitersuchen / Auswahl ... 333
12.14.6 Vorheriges suchen ... 333
12.14.7 Vorheriges suchen / Auswahl ... 333
12.15 Datei ... 334
12.15.1 Kategorie "Datei" ... 334
12.15.2 Projekteinstellungen - Benutzer und Gruppen ... 378
12.16 Hilfe ... 384
12.16.1 Befehle der Kategorie Hilfe ... 384
12.16.2 Inhalt ... 384
12.16.3 Index ... 384
12.16.4 Suchen ... 385
12.16.5 Informationen ... 385

12.16.6 Synchronisieren 385
12.17 Bildersammlung 386
12.17.1 Befehle zu Bildersammlung 386
12.17.2 Globale Bildersammlung erzeugen 386
12.17.3 Bildersammlung erzeugen 386
12.18 Installation 387
12.18.1 Bibliotheks-Repository 387
12.18.2 Visualisierungselemente-Repository 392
12.19 Bibliotheksverwalter 398
12.19.1 Befehle der Kategorie Bibliotheksverwalter 398
12.19.2 Bibliothek hinzufügen... 399
12.19.3 Eigenschaften... 403
12.19.4 Bibliothek erneut laden 404
12.20 Meldungsfenster 405
12.20.1 Befehle der Kategorie Meldungsfenster 405
12.20.2 Gehe zu Quelltextposition 405
12.20.3 Nächste Meldung 405
12.20.4 Vorherige Meldung 405
12.21 Objekte 406
12.21.1 Object Commands 406
12.21.2 Navigation 413
12.21.3 PLCopenXML 414
12.22 Online 417
12.22.1 Befehle der Kategorie Online 417
12.22.2 Einloggen 419
12.22.3 Ausloggen 422
12.22.4 Bootapplikation erzeugen 422
12.22.5 Laden 424
12.22.6 Online Change 424
12.22.7 Quellcode-Download... 425
12.22.8 Quellcode aus Steuerung laden... 426
12.22.9 Quellcode auf verbundene Steuerung schreiben 426

12.22.10 Start ... 427
12.22.11 Stop ... 427
12.22.12 Einzelzyklus ... 427
12.22.13 Mehrfacher Download ... 427
12.22.14 Reset Warm ... 430
12.22.15 Reset Kalt ... 430
12.22.16 Reset Ursprung ... 431
12.22.17 Werte schreiben ... 431
12.22.18 Werte forcen, Wert vorbereiten ... 432
12.22.19 Forcen aufheben ... 435
12.22.20 Alle Forces zum Überwachen-Fenster hinzufügen ... 435
12.22.21 Forceliste aufheben ... 436
12.22.22 Simulation ... 436
12.22.23 Ablaufkontrolle ... 437
12.22.24 Darstellung ... 441
12.22.25 Verbinden mit Gerät ... 441
12.22.26 Trennen von Gerät ... 441
12.22.27 Reset Ursprung Gerät ... 441
12.23 Einstellungen ... 441
12.23.1 Einstellungen ... 442
12.23.2 Optionen,»PLC Designer«-Konverter ... 457
12.23.3 Optionen, CFC Editor ... 459
12.23.4 Optionen, Deklarationseditor ... 460
12.23.5 Optionen, FUP, KOP und AWL ... 461
12.23.6 Optionen, Features ... 463
12.23.7 Optionen, Geräteeditor ... 466
12.23.8 Optionen, AS-Editor ... 467
12.23.9 Optionen, AS ... 470
12.23.10 Optionen, Quellcodeverwaltung ... 473
12.23.11 Optionen, Visualisierungen ... 474
12.23.12 Anpassen-Dialoge ... 476

12.24 Drucken 483
12.24.1 Befehle der Kategorie Drucken 483
12.24.2 Dokumentieren... 484
12.24.3 Seiteneinstellungen... 485
12.24.4 Drucken... 493
12.25 Rezepturverwalter 494
12.25.1 Rezepturverwalter Befehle 494
12.25.2 Rezeptur einfügen 494
12.25.3 Rezeptur entfernen 495
12.26 Intelligente Code-Bearbeitung 495
12.26.1 Eingabehilfe 495
12.27 AS 500
12.27.1 AS-Editor Befehle 500
12.27.2 Initialschritt 501
12.27.3 Schritt-Transition einfügen 501
12.27.4 Schritt-Transition danach einfügen 502
12.27.5 Parallel 502
12.27.6 Alternativ 503
12.27.7 Verzweigung einfügen 503
12.27.8 Verzweigung rechts einfügen 503
12.27.9 Aktionsassoziation einfügen 505
12.27.10 Aktionsassoziation danach einfügen 505
12.27.11 Sprung einfügen 506
12.27.12 Sprung danach einfügen 506
12.27.13 Makro einfügen 507
12.27.14 Makro danach einfügen 507
12.27.15 Makro anzeigen 507
12.27.16 Makro verlassen 508
12.28 Quellcodeverwaltung 509
12.28.1 Befehle der Kategorie Quellcodeverwaltung 509
12.28.2 Quellcodeverwaltung ändern 510
12.28.3 Projekt aus Quellcodeverwaltung öffnen... 514

12.28.4 Objekte aus der Quellcodeverwaltung hinzufügen 515
12.28.5 Letze Version abrufen - Gesamtes Projekt 516
12.28.6 Letzte Version abrufen - Ausgewählte Objekte 516
12.28.7 Auschecken... 516
12.28.8 Einchecken... 518
12.28.9 Auschecken rückgängig 519
12.28.10 Unterschiede anzeigen... 519
12.28.11 Versionsgeschichte... 520
12.28.12 Projektversionsgeschichte... 522
12.28.13 Bezeichnung... 524
12.28.14 Status aktualisieren 524
12.28.15 Anmelden... 525
12.29 Text Editor 526
12.29.1 Befehle der Kategorie Texteditor 526
12.29.2 Überschreibmodus 526
12.29.3 Gehe zu Zeile... 526
12.29.4 Datei als Text einfügen... 526
12.29.5 In Großbuchstaben umwandeln 527
12.29.6 In Kleinbuchstaben umwandeln 527
12.29.7 Gehe zur zugehörigen Klammer 527
12.29.8 Bis zur zugehörigen Klammer selektieren 527
12.30 Textlisten 528
12.30.1 Befehle der Kategorie Textlisten 528
12.30.2 Globale Textliste anlegen 528
12.30.3 Sprache einfügen 529
12.30.4 Sprache entfernen 529
12.30.5 Import/Export Textlisten 530
12.30.6 Visualisierungs-Text-IDs aktualisieren 535
12.30.7 Visualisierungs-Text-IDs prüfen 535
12.30.8 Nicht verwendete Textlisteneinträge entfernen 535

12.31 Trace 536
12.31.1 Befehle der Kategorie Trace 536
12.31.2 Trace einloggen 536
12.31.3 Start/Stop Trace 537
12.31.4 Trigger zurücksetzen 537
12.31.5 Cursor 537
12.31.6 Scrollen 538
12.31.7 Zoomen 539
12.31.8 Anzeige zurücksetzen 539
12.31.9 Dehnen 539
12.31.10 Komprimieren 540
12.31.11 Mehrfachkanal 540
12.31.12 Trace laden... 540
12.31.13 Trace speichern... 540
12.31.14 Trace Tastenkombinationen 541
12.32 Rückgängig/Wiederherstellen 542
12.32.1 Befehle der Kategorie Rückgängig/Wiederherstellen 542
12.32.2 Rückgängig 542
12.32.3 Wiederherstellen 542
12.33 Benutzerverwaltung 543
12.33.1 Befehle der Kategorie Benutzerverwaltung 543
12.33.2 Benutzer anmelden 543
12.33.3 Benutzer abmelden 545
12.33.4 Rechte... 546
12.34 Visualisierungskommandos 548
12.34.1 Befehle der Kategorie Visualisierungskommandos 548
12.34.2 Visualisierungselement einfügen 549
12.34.3 Alles selektieren 549
12.34.4 Alles deselektieren 549
12.34.5 Gruppieren 549
12.34.6 Gruppierung aufheben 549
12.34.7 Frame Auswahl 550

12.34.8 Schnittstellen-Editor ... 551
12.34.9 Tastaturbedienung ... 551
12.34.10 Elementliste ... 552
12.34.11 Hintergrund ... 552
12.34.12 Reihenfolge ... 552
12.34.13 Ausrichtung ... 553
12.34.14 Tastaturbedienung aktivieren ... 554
12.35 Ansicht ... 554
12.35.1 Querverweisliste ... 554
12.35.2 Haltepunkte ... 557
12.35.3 Standardansichten ... 560
12.35.4 PLC Designer Frame ... 561
12.35.5 Meldungen ... 580
12.35.6 AS Object ... 581
12.35.7 Quellcodeverwaltung ... 581
12.35.8 Werkzeuge ... 583
12.35.9 Visualisierungselement ... 583
12.35.10 Überwachen ... 584
12.36 Fenster ... 584
12.36.1 Fenster Befehle - »PLC Designer« Frame ... 584
12.36.2 Andocken ... 585
12.36.3 Unverankert ... 585
12.36.4 Ausblenden ... 585
12.36.5 Nächster Editor ... 585
12.36.6 Vorheriger Editor ... 585
12.36.7 Nächstes Teilfenster ... 586
12.36.8 Vorheriges Teilfenster ... 586
12.36.9 Editor schließen ... 586
12.36.10 Alle Editoren schließen ... 586
12.36.11 Fenster-Layout zurücksetzen ... 586
12.36.12 Neue horizontale Registerkartengruppe ... 587
12.36.13 Neue vertikale Registerkartengruppe ... 587

12.36.14 Fenster <n> 587
12.36.15 Fenster 587

13 Allgemeine Editoren 588

13.1 CAM Editor 588
13.1.1 Eine kurze Übersicht 588
13.1.2 Definition einer SoftMotion Kurvenscheibe 588
13.1.3 Anlegen einer Kurvenscheibe 590
13.1.4 Editieren einer Kurvenscheibe 591
13.1.5 Kompilieren der Kurvenscheibendefinition 598
13.1.6 Allgemeine Hinweise zu Kurvenscheiben 598
13.1.7 CAM Datenstrukturen 607
13.1.8 CAM: Elementeigenschaften 611

13.2 CFC-Editor 613
13.2.1 CFC - Continuous Function Chart - Sprache 614
13.2.2 Cursor-Positionen in CFC 614
13.2.3 CFC Elemente / Werkzeuge 616
13.2.4 Elemente einfügen und anordnen 621
13.2.5 CFC Editor im Online-Modus 623

13.3 CNC-Editor 626
13.3.1 Texteditor 633
13.3.2 CNC-Tabelleneditor 673
13.3.3 Grafischer Editor 675
13.3.4 Eigenschaften Bahnelemente 679
13.3.5 Werkzeuge 679
13.3.6 CNC-Objekt 683
13.3.7 CNC-Datenstrukturen 688
13.3.8 CNC Online 690

13.4 CNC-Einstellungen-Editor 690
13.4.1 Parameter-Editor 693

13.5 Datenquellen-Editor ... 694
13.5.1 Data Server, Überblick ... 694
13.5.2 Data Server hinzufügen ... 695
13.5.3 Datenquelle hinzufügen ... 696
13.5.4 Datenquelle konfigurieren ... 697
13.5.5 Datenquellen in Visualisierungen verwenden ... 704
13.6 Deklarationseditor ... 705
13.7 DUT-Editor ... 710
13.8 FUP/KOP/AWL-Editor ... 710
13.8.1 Programmiersprachen ... 711
13.8.2 Modifikatoren und Operatoren in AWL ... 713
13.8.3 Arbeiten im FUP- und KOP-Editor ... 716
13.8.4 Arbeiten im AWL-Editor ... 720
13.8.5 Cursor-Positionen in FUP, KOP und AWL ... 727
13.8.6 FUP/KOP/AWL Menü ... 730
13.8.7 FUP/KOP/AWL Elemente ... 731
13.8.8 FUP/KOP/AWL Editor im Online-Modus ... 742
13.9 Globale-Variablenlisten-Editor ... 747
13.10 Bibliotheksverwalter ... 747
13.10.1 Editorfenster des Bibliotheksverwalters ... 748
13.10.2 Bibliotheksmenü ... 751
13.11 Netzwerkvariablenlisten Editor ... 751
13.12 Rezepturverwalter ... 752
13.12.1 Rezepturdefinition ... 754
13.12.2 RecipeManCommands ... 758
13.12.3 Rezepturverwalter im Online-Modus ... 764
13.13 AS-Editor ... 764
13.13.1 AS - Ablaufsprache ... 766
13.13.2 Cursor-Positionen in AS ... 767
13.13.3 Arbeiten im AS-Editor ... 768
13.13.4 AS Element-Eigenschaften ... 771
13.13.5 AS Elemente / Toolbox ... 772

13.13.6 Qualifizierer für Aktionen in AS ... 783
13.13.7 Implizite Variablen - AS Flags ... 784
13.13.8 Abarbeitungsreihenfolge in AS ... 790
13.13.9 AS-Editor im Online-Modus ... 792
13.14 ST-Editor ... 794
13.14.1 ST-Editor im Online-Modus ... 794
13.14.2 Strukturierter Text ST / ExST ... 797
13.15 Symbolkonfigurationseditor ... 809
13.16 Taskkonfigurationseditor ... 811
13.16.1 Task Editor, Verwendung ... 812
13.16.2 Task Editor im Online-Modus ... 818
13.17 Trace-Editor ... 820
13.17.1 Trace-Editor ... 820
13.18 Visualisierungseditor ... 838
13.18.1 Visualisierungselemente einfügen, Werkzeugbox ... 839
13.18.2 Elemente selektieren ... 841
13.18.3 Position, Größe, Ausrichtung, Reihenfolge ... 842
13.18.4 Gruppierung von Elementen ... 842
13.18.5 Visualisierungselemente - Eigenschaften ... 843
13.18.6 Schnittstellen-Editor für Visualisierung ... 884
13.18.7 Tastaturkonfigurations-Editor ... 888
13.18.8 Elementliste ... 890
13.19 Visualisierungsmanager ... 891
13.19.1 Target-Visualisierung ... 895
13.19.2 Web-Visualisierung ... 896
13.20 Überwachen-Editor ... 897
13.20.1 Überwachungsliste anlegen ... 897
13.20.2 Überwachungsliste im Online-Modus ... 898

14 Geräte-Editoren 900

14.1 Generischer Geräte-Editor 900

14.2 CANopen-Konfigurationseditor 924

14.2.1 CANbus 924

14.3 EtherCAT-Konfigurationseditor 948

14.4 Profibus DP-Konfigurationseditor 977

14.5 SoftMotion-Geräteeditor 989

14.5.1 SoftMotion Konzept und Übersicht über Komponenten 989

14.5.2 Das SoftMotion Drive Interface 992

14.5.3 Konfiguration mit Hilfe des SoftMotion Geräte-Editors 993

14.5.4 Konfiguration Drive Logical 997

14.5.5 Übersicht über spezielle Anwendungsfälle 999

14.5.6 Beispielapplikationen 1001

15 Kommandozeilen-Optionen 1024

15.1 Starten von PLC Designer.exe 1024

15.1.1 --culture 1024

16 Programmierreferenz 1025

16.1 Deklarieren 1025

16.1.1 Variablendeklaration 1025

16.1.2 Empfehlungen für die Vergabe von Bezeichnern 1028

16.1.3 Variablen-Initialisierung 1034

16.1.4 Beliebige Ausdrücke für Variablen-Initialisierung 1035

16.1.5 Deklarationseditor 1035

16.1.6 Autodeklarations-Funktion 1035

16.1.7 Kurzform-Modus 1035

16.1.8 AT-Deklaration 1037

16.1.9 Schlüsselwörter 1038

16.1.10 Lokale Variablen - VAR 1042

16.1.11 Eingabevariablen - VAR_INPUT 1042

16.1.12 Ausgabevariablen - VAR_OUTPUT 1042

16.1.13 EinAusgabevariablen - VAR_IN_OUT 1043

16.1.14 Globale Variablen - VAR_GLOBAL ... 1043
16.1.15 Temporäre Variablen - VAR_TEMP ... 1044
16.1.16 Statische Variablen - VAR_STAT ... 1044
16.1.17 Externe Variablen - VAR_EXTERNAL ... 1044
16.1.18 Attribut-Schlüsselwörter für Variablentypen ... 1045
16.1.19 Remanente Variablen - RETAIN, PERSISTENT ... 1045
16.1.20 Konstanten - CONSTANT, Getypte Konstanten ... 1048
16.1.21 Variablenkonfiguration - VAR_CONFIG ... 1049
16.1.22 Benutzerdefinierte Datentypen ... 1051
16.1.23 Erweiterbare Funktionen - PARAMS ... 1051
16.1.24 FB_init, FB_reinit Methoden ... 1051
16.1.25 FB_exit Methode ... 1053
16.1.26 Pragma Anweisungen ... 1054
16.1.27 Komponenten Auflisten Funktion ... 1087
16.2 Datentypen ... 1088
16.2.1 Standard Datentypen ... 1088
16.2.2 Erweiterungen zur Norm IEC 61131-3 ... 1090
16.2.3 Benutzerdefinierte Datentypen ... 1096
16.3 Operatoren ... 1111
16.3.1 IEC Operatoren und norm-erweiternde Funktionen ... 1111
16.3.2 Arithmetische Operatoren ... 1111
16.3.3 Bitstring Operatoren ... 1117
16.3.4 Bit-Shift Operatoren ... 1120
16.3.5 Auswahloperatoren ... 1124
16.3.6 Vergleichsoperatoren ... 1127
16.3.7 Adressoperatoren ... 1131
16.3.8 Aufrufoperatoren ... 1132
16.3.9 Typkonvertierungs-Operatoren ... 1133
16.3.10 Numerische Operatoren ... 1142
16.3.11 IEC-erweiternde Funktionen ... 1148

16.4 Operanden 1162
16.4.1 Konstanten 1162
16.4.2 Variablen 1167
16.4.3 Adressen 1169
16.4.4 Funktionen 1172

17 Geräte 1173
17.1 L-force Controller 3200 C 1173
17.1.1 Automatisches Erkennen der I/O-Komplettmodul-Topologie 1177
17.1.2 Zykluszeit des Zugriffs auf die I/O-Komplettmodule einstellen 1178
17.1.3 Rechenzeitbedarf einzelner SoftMotion Funktionsbausteine 1179

18 PLC-Designer Bibliotheken 1180

19 Index 1182

1 Über diese Dokumentation

Diese Dokumentation enthält Informationen zur SPS-Programmierung mit dem »PLC Designer« und erleichtert Ihnen den Einstieg in die mächtigen Sprachmittel der IEC 61131.

Zielgruppe

Diese Dokumentation richtet sich an qualifiziertes Fachpersonal nach IEC 364.

Informationen zur Gültigkeit

Die Informationen in dieser Dokumentation sind gültig für folgende Lenze-Software:

Software	ab Softwarestand
L-force »PLC Designer«	3.0.0

Dokumenthistorie

Version			Beschreibung
1.0	06/2008	TD29	Erstausgabe für den »PLC Designer«
2.0	08/2008	TD29	Überarbeitung
3.0	05/2009	TD29	Überarbeitung
4.0	09/2010	TD29	Aktualisierung auf Software V3.4
4.1	03/2011	TD29	Aktualisierung auf Software V3.5

Informationen und Hilfsmittel rund um die Lenze Produkte finden Sie im Download-Bereich unter: http://www.Lenze.com

Haftung

Wir haben alle Angaben in dieser Dokumentation mit größter Sorgfalt zusammengestellt und auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Trotzdem können wir Abweichungen nicht ganz ausschließen. Wir übernehmen keine juristische Verantwortung oder Haftung für Schäden, die dadurch eventuell entstehen. Notwendige Korrekturen werden wir in die nachfolgenden Auflagen einarbeiten.

Diese Dokumentation basiert auf der original »CoDeSys«-Dokumentation der Firma 3S - Smart Software Solutions GmbH (Kempten). Die dargestellten Screenshots können von der »PLC Designer«-Darstellung abweichen.

Warenzeichen

Microsoft, Windows und Windows NT sind entweder eingetragene Warenzeichen oder Warenzeichen der Microsoft Corporation in den USA und/oder anderen Ländern.

Adobe und Reader sind entweder eingetragene Warenzeichen oder Warenzeichen von Adobe Systems Incorporated in den USA und/oder anderen Ländern.

Alle anderen in dieser Dokumentation aufgeführten Markennamen sind Warenzeichen ihrer jeweiligen Besitzer.

2 Lenze Spezifika

Der »PLC Designer« basiert auf der IEC-Entwicklungsumgebung »CoDeSys« (Controller Development System) der Firma 3S - Smart Software Solutions GmbH, Kempten.

Im Gegensatz zum »CoDeSys« unterstützt der »PLC Designer« folgende Funktionen/Hardware **nicht**:

- CanDevice (CANopen Slave)
- DeviceNet-Module
- Encoder- und Sercos-Interfaces
- Hardware Scan/Status/Diagnose aus dem Zielsystem
- »PLC Designer« HMI
- »PLC Designer«-Target-Visualisierung

2.1 Lenze-Funktionsbausteine und Funktionen

2.1.1 Bibliothek: L_PLCLoggingAccess.lib

Funktion L_AddLog - Logbucheintrag erzeugen

Funktionsbibliothek: L_PLCLoggingAccess.lib

Diese Funktion ermöglicht es, einen Eintrag in das in das Logbuch des L-force Controllers vorzunehmen. Sie liefert den Rückgabewert TRUE, wenn der Eintrag vorgenommen wurde.

Eingänge

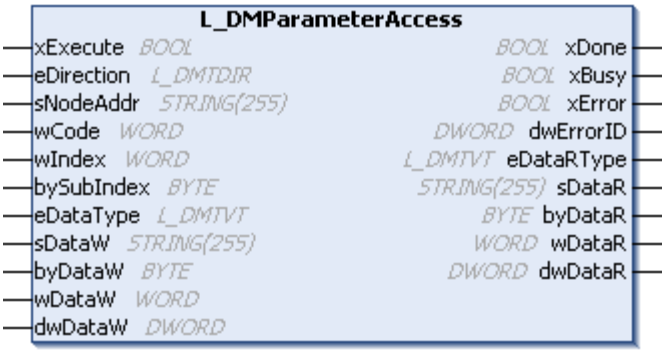
Bezeichner / Datentyp	Info / Einstellmöglichkeiten
eSeverity / L_TSeverity	Logging Severity - Risiko des Fehlers L_FatalError L_Error, L_Warning L_Information Der Enum Parameter eSeverity (IEC_INT) wird als Severity (BYTE) an Add2Log übergeben.
eFacility / L_TFacility	Logging Facility: - Bereich aus dem der Fehler gemeldet wird L_PLC_USER_LOG_SCOPE1 L_PLC_USER_LOG_SCOPE2 L_PLC_USER_LOG_SCOPE3 Der Enum Parameter eFacility (IEC_INT) wird als Facility (WORD) an Add2Log übergeben.
sLogPayLoad / STRING(80)	Logbucheintrag Der Ansi String aus "sLogPayLoad" wird als 1-byte-String-Parameter an Add2Log übergeben. ("hs", SLOGPAYLOAD)

2.1.2 Bibliothek: PLCDataManagerAccess.lib

Funktionsbaustein L_SMC_ParameterAccess_FB

Funktionsbibliothek: PLCDataManagerAccess.lib

Mit dem Funktionsbaustein L_SMC_ParameterAcces_FB können die Geräteparameter des L-force Controllers von der PLC gelesen und geschrieben werden.



Hinweis: Der PLC-Baustein L_SMC_ParameterAccess_FB unterstützt keine Codestellen die Float/Double-Parameter enthalten (z. B. Versorgungsspannung C00205 oder Akku-Spannung C00207).

Wird dieser Baustein dennoch auf solche Codestellen angewendet, wird die Fehlernummer 532 ("error if the format from the code number is not BYTE, WORD, DWORD or BSTR") gemeldet.

Eingänge

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
xExecute / BOOL	steigende Flanke startet den Zugriff
eDirection / L_DMTDIR	Zugriffsrichtung L_DM_READ - lesen L_DM_WRITE - schreiben
sNodeAddr / STRING(255)	Topologie-Adresse '/' - Zugriff auf Parameter des L-force Controllers
wCode / WORD	Code-Stelle des Parameters (Lenze) dann muss wIndex zu 0 gesetzt sein
wIndex / WORD	Index des Parameters (CANopen) dann muss wCode zu 0 gesetzt sein
bySubIndex / BYTE	Subindex/Subcode-Stelle des Parameters
eDataType / L_DMTVT	Angabe des Typs des zu schreibenden Parameters {L_DM_VT_EMPTY, LDM_VT_I4, ...}
sDataW / STRING(255)	zu schreibender Wert (string) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
byDataW / BYTE	zu schreibender Wert (1*byte) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"
wDataW / WORD	zu schreibender Wert (2*byte) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"
dwDataW / WORD	zu schreibender Wert (4*byte) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"

Ausgänge

Bezeichne / Datentyp	Wert / Bedeutung
bDone / BOOL	TRUE wenn Operation erfolgreich beendet
bBusy / BOOL	TRUE solange die Operation fehlerfrei läuft
bError / BOOL	TRUE wenn ein Fehler aufgetreten ist
dwErrorID / DWORD	Größer 0 wenn ein Fehler aufgetreten ist
eDataRType / L_DMVT	zurückgegebener Typ des gelesenen Parameters {L_DM_VT_EMPTY, LDM_VT_I4, ...}
sDataR / STRING(255)	gelesener Wert (string) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"
byDataR / BYTE	gelesener Wert (1*byte) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"
wDataR / WORD	gelesener Wert (2*byte) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"
dwDataR / DWORD	gelesener Wert (4*byte) - Tabelle "Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp"

Zusammenhang zwischen den Ein- und Ausgängen und dem Datentyp

Eingang, Ausgang	Datentyp
sDataR, sDataW	L_DM_VT_BSTR
byDataR, byDataW	L_DM_VT_I1, L_DM_VT_UI1
wDataR, wDataW	L_DM_VT_I2, L_DM_VT_UI2
dwDataR, dwDataW	L_DM_VT_I4, L_DM_VT_UI4, L_DM_VT_DATE

Fehlermeldungen vom Funktionsbaustein

Fehlernummer	Bedeutung
507	Fehlerhafte Formatangabe bei einem Schreibzugriff
531	Fehlerhafte Formatangabe bei der Beauftragung
532	Codestelle enthält Float/Double-Parameter (z. B. Versorgungsspannung C00205 oder Akku-Spannung C00207); dieses Format wird vom Baustein nicht unterstützt.
weitere	Interner Software-Fehler; wenden Sie sich an den Lenze-Service.

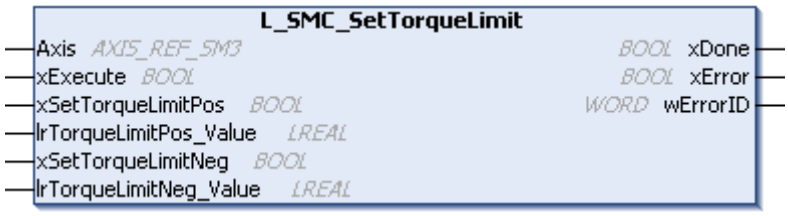
2.1.3 Bibliothek: L_SM3_DriveUtil.lib

Funktionsbaustein L_SMC_SetTorqueLimit (für EtherCAT)

Funktionsbibliothek: L_SM3_DriveUtil.lib

Die Drehmomentgrenzwerte eines Antriebes in positiver und in negativer Drehrichtung lassen sich von der Steuerung vorgeben. Bisher erfolgte das Setzen der Grenzwerte über SDO-Zugriffe. Bei Ansteuerung der Antriebe über einen EtherCAT-Feldbus ist es möglich, diese Werte als Teil der zyklischen Prozessdaten mit zu übertragen. Der Vorteil hierbei ist die wesentlich schnellere Übertragung der Werte zum Antrieb. Um dies nutzen zu können, muss im EtherCAT Konfigurator ein entsprechendes Mapping ausgewählt werden, welches diese Drehmomentgrenzwerte als Prozessdatum enthält.

Der Zugriff auf die Drehmomentgrenzwerte aus der PLC erfolgt über den Funktionsbaustein L_SMC_SetTorqueLimit:



Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Execute / BOOL	Aktiviert die Funktion.
SetTorqueLimitPos / BOOL	Beauftragt einen positiven Drehmomentgrenzwert
TorqueLimitPos_Value / LREAL	Positiver Drehmomentgrenzwert im Bereich 0 ... 100 % des Nennmoments
SetTorqueLimitNeg / BOOL	Beauftragt einen negativen Drehmomentgrenzwert
TorqueLimitNeg_Value / LREAL	Negativer Drehmomentgrenzwert im Bereich 0 ... 100 % des Nennmoments
Axis / AXIS_REF	Hier wird die Achs-Struktur übergeben, für welche diese Drehmomentgrenzwerte gelten.

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
Done / BOOL	TRUE zeigt an, dass die Vorgabe quittiert ist.
Error / BOOL	TRUE zeigt an, dass im Funktionsbaustein ein Fehler aufgetreten ist.
ErrorID / INT	Fehlernummer

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlermeldung	Beschreibung
ERROR_NO_LENZE_AXIS_REF: INT := 16#200	Funktion wird von diesem Antrieb nicht unterstützt.
ERROR_TORQUE_LIMIT_NOT_VALID: INT := 16#210	Wertebereich überschritten. Nur Drehmomentgrenzwerte von 0 ... 200 % sind zulässig.
ERROR_TORQUE_LIMIT_MAPPING_FAULT := 16#211	Diese Funktion ist mit dem angewählten Mapping nich ausführbar. Es fehlen die Drehmoment-Prozessdaten.

Beispiel 1: Die PLC will den Drehmomentgrenzwert positiv auf 50 % setzen.

SetTorqueLimitPos = TRUE
TorqueLimitPos_Value: 50.0
SetTorqueLimitNeg = FALSE
TorqueLimitNeg_Value = xxx
Execute = True

Der Antrieb setzt den positiven Drehmomentgrenzwert auf 50 % des Nennmomentes; der im Antrieb gültige negative Drehmomentgrenzwert wird nicht verändert.

Beispiel 2: Die PLC will beide Drehmomentgrenzwerte auf 100 % setzen.

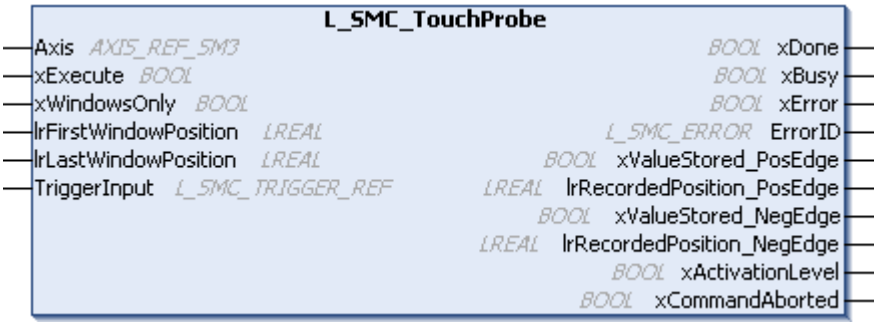
SetTorqueLimitPos = TRUE
TorqueLimitPos_Value: 100.0
SetTorqueLimitNeg = TRUE
TorqueLimitNeg_Value = 100.0
Execute = True

Der Antrieb setzt beide Drehmomentgrenzwerte auf 100 % des Nennmoments.

Funktionsbaustein L_SMC_TouchProbe (für EtherCAT)

Funktionsbibliothek: L_SM3_DriveUtil.lib

Dieser Baustein dient dazu, über einen schnellen Eingang die Position des Antriebs hoch-genau zu erfassen. Bisher erfolgte die Beauftragung und Rückmeldung von Messwerten über SDO-Zugriffe. Bei Ansteuerung der Antriebe über einen EtherCAT-Feldbus ist es möglich, diese Daten als Teil der zyklischen Prozessdaten mit zu übertragen. Der Vorteil hierbei ist die wesentlich schnellere Übertragung der Daten von und zum Antrieb. Um dieses nutzen zu können, müssen Sie ein entsprechendes Mapping auswählen, welches diese Daten als Prozessdatum enthält.



Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Execute / BOOL	Mit Setzen einer neuen positiven Flanke wird eine neue Messung gestartet. Alle Ausgänge des Bausteins werden auf die Initialwerte zurückgesetzt.
WindowsOnly / BOOL	Fensterauswertung aktivieren: Alle Messwerte außerhalb des definierten Fensters werden ausgeblendet.
FirstWindowsPosition / LREAL	Untere Grenze des Fensters in SoftMotion Einheiten [SM-Einheiten]
LastWindowPosition / LREAL	Obere Grenze des Fensters in SoftMotion Einheiten [SM-Einheiten]
TriggerInput / L_SMC_Trigger_REF	Siehe nachfolgende Beschreibung
Axis / AXIS_REF	Auswahl des Antriebes

TriggerInput / L_SMC_Trigger_REF:

Variable	Typ	Initialwert	Beschreibung
TouchProbeSelect	Byte	1	Auswahl TP1 oder TP2 1: TP_Kanal1 -> 9400; DigIn 7 oder Nullimpuls des Lagegebers 2: TP_Kanal2 -> 9400; DigIn 8 oder Nullimpuls Lagegeber
TouchProbeMode	Byte	1	Auswahl Betriebsart 1: Einzelmessung; Execute-Flanke aktiviert die Funktion bis Done = 1 (TRUE) oder Abbruch durch FB L_SMC_AbortTrigger. 2: Kontinuierliche Messung; positive Execute-Flanke aktiviert die Funktion bis negative Execute-Flanke (Execute = FALSE) oder Abbruch durch FB L_SMC_AbortTrigger.
TouchProbeSignal	Byte	1	Auswahl Nullimpuls oder Digitaler Eingang 1: Digitaler Eingang 2: Nullimpuls des Lagegebers
TouchProbeEdge	Byte	1	Konfiguration der Flankenauswertung 1: Triggern mit positiver Flanke 2: Triggern mit negativer Flanke 3: Triggern mit positiver oder negativer Flanke (Logik im FB) 4: Ohne Funktion. 5: Triggern mit positiver und negativer Flanke (Die Reihenfolge der Flanken ist relevant (Logik im Antrieb).) 6: Triggern mit negativer und positiver Flanke (Die Reihenfolge der Flanken ist relevant (Logik im Antrieb))

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
Done / BOOL	In der Betriebsart "Einzelmessung": TRUE = Messvorgang abgeschlossen. In der Betriebsart "Kontinuierliches Messen": Ohne Funktion (Siehe Eingangsvariable aus TriggerInput / L_SMC_Trigger_REF "TouchProbeMode")
Busy / BOOL	Baustein aktiviert, warten auf Trigger.
Error / BOOL	Fehlerkennung: Ein Fehler ist in der Beauftragung oder in der Kommunikation mit dem Antrieb aufgetreten.
ErrorID / INT	Fehlernummer
ValueStored_PosEdge / BOOL	Messwert gespeichert nach Triggern auf positive Flanke.
RecordedPosition_PosEdge / LREAL	Messwert beim Triggern mit der positiven Flanke in SoftMotion-Einheiten [SM-Einheiten], d. h. der Messwert wird in der selben Einheit an die PLC-Applikation geliefert, in der die Achse auch programmiert wird.
ValueStored_NegEdge / BOOL	Messwert gespeichert nach Triggern auf negative Flanke.
RecordedPosition_NegEdge / LREAL	Messwert beim Triggern mit der negativen Flanke in SoftMotion-Einheiten [SM-Einheiten], d. h. der Messwert wird in der selben Einheit an die PLC-Applikation geliefert, in der die Achse auch programmiert wird.
ActivationLevel / BOOL	Pegel des TP-Eingangs zum Zeitpunkt der TP-Funktionsaktivierung im Antrieb. Die Applikation kann durch Auswertung des ActivationLevel bei der Beauftragung "positive oder negative Flanke" die Art der ersten aktiven Flanke erkennen.
CommandAborted / BOOL	Baustein abgebrochen über L_SMCAbortTrigger. Die Ausgänge "Busy" und "Done" werden zurückgesetzt.

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlernummer	Beschreibung
411	TouchProbeSelect fehlerhaft: TP-Kanal nicht vorhanden TouchProbeSelect > 1 bei ECSxM TouchProbeSelect > 2 bei 9400
412	TouchProbeMode fehlerhaft: TouchProbeMode > 1 bei ECSxM (Es wird nur 'Einzelmessung' unterstützt.) TouchProbeMode > 2 bei 9400
413	TouchProbeEdge fehlerhaft: Bei Auswahl "TouchProbeSignal = Nullimpuls" nur Flankenauswertung 'positive Flanke' möglich TouchProbeMode > 1 bei ECSxM (Es wird nur 'Einzelmessung' unterstützt.) TouchProbeMode > 2 bei 9400
414	TouchProbeEdge fehlerhaft: Angeforderte Flankenauswertung nicht vorhanden TouchProbeEdge > 2 bei ECSxM (Nur positive oder negative Flanke möglich.) TouchProbeEdge > 4 bei 9400 (Art der Flankenauswertung wird nicht unterstützt.)
415	TP-Kanal nicht frei: TP-Kanal wurde bereits über anderen Baustein aktiviert.
416	Touch Probe ist bei aktuell angewähltem Prozessdaten-Mapping nicht möglich.
417	Interner Fehler / TP-Kanal nicht frei: TP-Kanal wurde bereits über anderen Baustein aktiviert.
418	Interner Fehler: Timeout bei der Beauftragung des Antriebes

Funktionsbaustein L_SMC_AbortTrigger (für EtherCAT)

Funktionsbibliothek: L_SM3_DriveUtil.lib

Hinweis: Dieser Baustein unterstützt zur Zeit nur EtherCAT-Achsen.

Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Execute / BOOL	Mit dem Setzen einer neuen positiven Flanke werden alle Ausgänge des Bausteins auf die Initialwerte zurückgesetzt.
TriggerInput / L_SMC_Trigger_REF	Struktur zur Konfiguration der Messung
Axis / AXIS_REF	Auswahl des Antriebes

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
Done / BOOL	Funktionsbaustein L_SMC_TouchProbe wurde abgebrochen.
Error / BOOL	Fehlerkennung: Ein Fehler ist in der Beauftragung aufgetreten.
ErrorID / INT	Fehlernummer

Fehlermeldungen

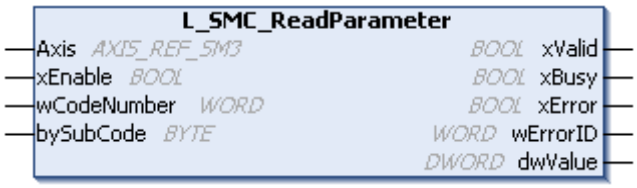
Folgende Fehlermeldungen sind hinterlegt:

Fehlernummer [hex]	Beschreibung
0x0001	Fehler in Beauftragung des Bausteins: Eingangsdaten stimmen nicht mit aktivierter TP-Beauftragung auf diesem Kanal überein. Ist auf dem TP-Kanal kein Auftrag aktiv, wird keine Fehlermeldung erzeugt. TouchProbeSelect > 1 bei ECSxM TouchProbeSelect > 2 bei 9400

Funktionsbaustein L_SMC_ReadParameter (für EtherCAT)

Funktionsbibliothek: L_SM3_DriveUtil.lib

Mit diesem Baustein werden Codestellen eines Motion-Antriebes ausgelesen. Codestelle und Subindex werden hierbei direkt am Baustein angegeben. Das Umrechnen auf die entsprechende Parameternummer erfolgt intern.



Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Enable / BOOL	Aktiviert die Funktion.
wCodeStelle / WORD	Angabe der Codestelle die gelesen werden soll.
bySubIndex / BYTE	Angabe des Subindex der gelesen werden soll.
Axis / AXIS_REF	Hier wird die Achs-Struktur übergeben, für welche die Codestelle gelesen werden soll.

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
Done / BOOL	TRUE zeigt an, dass die Vorgabe quittiert ist.
Error / BOOL	TRUE zeigt an, dass im Funktionsbaustein ein Fehler aufgetreten ist
ErrorID / INT	Fehlernummer
dwValue / DWORD	Gelesener Wert

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlernummer [hex]	Beschreibung
SMC_RP_PARAM_UNKNOWN:=130	Unbekannte Codestelle
SMC_RP_REQUESTING_ERROR:=131	Leseauftrag wurde abgebrochen, z. B. weil der Antrieb in der vorgegebenen Zeit nicht geantwortet hat.

Funktionsbaustein L_SMC_WriteParameter (für EtherCAT)

Funktionsbibliothek: L_SM3_DriveUtil.lib

Mit diesem Baustein werden Codestellen eines Motion-Antriebes geschrieben. Codestelle und Subindex werden hierbei direkt am Baustein angegeben. Das Umrechnen auf die entsprechende Parameternummer erfolgt intern.

Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Execute / BOOL	Aktiviert die Funktion.
wCodeStelle / WORD	Angabe der Codestelle in die geschrieben werden soll.
bySubIndex / BYTE	Angabe des Subindex in den geschrieben werden soll.
dwValue / DWORD	Wert, der geschrieben werden soll.
Axis / AXIS_REF	Hier wird die Achs-Struktur übergeben, für welche die Codestelle geschrieben werden soll.

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
Done / BOOL	TRUE zeigt an, dass die Vorgabe quittiert ist.
Error / BOOL	TRUE zeigt an, dass im Funktionsbaustein ein Fehler aufgetreten ist
ErrorID / INT	Fehlernummer

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlernummer [hex]	Beschreibung
SMC_WP_PARAM_UNKNOWN:=140	Unbekannte Codestelle
SMC_WP_SENDING_ERROR:=141	Schreibauftrag wurde abgebrochen, z. B. weil der Antrieb in der vorgegebenen Zeit nicht geantwortet hat.

Funktionsbaustein L_SMC_ReadAxisError

Funktionsbibliothek: L_SM3_DriveUtil.lib

Mit diesem Baustein wird die Fehler-Codestelle eines Motion-Antriebes ausgelesen.

Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Axis / AXIS_REF	Hier wird die Achs-Struktur übergeben, für welche die Codestelle gelesen werden soll.
xEnable / BOOL	Aktiviert die Funktion.

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
xValid / BOOL	Auftrag wurde ausgeführt
xBusy / BOOL	Auftrag gestartet
xError / BOOL	Auftrag mit Fehler ausgeführt
wErrorID / WORD	Fehlernummer bei dieser Beauftragung, nicht die Fehlernummer des Antriebes
xAxisError / BOOL	Antrieb hat einen Fehler gemeldet
dwAxisErrorID / DWORD	Gelesene Fehlernummer des Antriebes

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlernummer [hex]	Beschreibung
SMC_RP_PARAM_UNKNOWN:=130	Unbekannte Codestelle
SMC_RP_REQUESTING_ERROR:=131	Leseauftrag wurde abgebrochen, z. B. weil der Antrieb in der vorgegebenen Zeit nicht geantwortet hat.

Die Bedeutung der Antriebsfehlernummer entnehmen Sie bitte der Antriebsdokumentation.

2.1.4 Bibliothek: L_SM3_DriveUtil.lib (I/O-System 1000)

Grundlegende Funktionen der schnellen Ein-/Ausgänge

Die im Folgenden beschriebenen I/O-Komplettmodule gehören zum I/O-System 1000, welches für die Realisierung komplexer Automatisierungsanwendungen geeignet ist. Das System besteht aus einem Buskopplermodul und diversen I/O–Komplettmodulen, die über einen internen Rückwandbus miteinander stationsintern kommunizieren, sowie Prozessdaten, Parameterdaten und Diagnoseinformationen austauschen.

2 digitale Eingänge mit Time Stamp-Funktion - EPM–S207

Dieser "schnelle Eingang" erfasst bis zu zwei binäre Steuersignale von der Prozessebene und überträgt diese zum übergeordneten Bussystem. Über einen internen 16-Bit-Zähler (µs–Ticker) kann der Zeitpunkt (Latch-Wert) registriert werden, wann ein bestimmtes Ereignis stattgefunden hat (Time Stamp-Funktionalität).

Schnellen Eingänge werden eingesetzt für:

- Diskontinuierliche Bewegung auf Bahnmarke ausrichten
  Fliegende Säge / Rotierendes Messer / Vorschub
- Kontinuierliche Bewegung auf Bahnmarke ausrichten
  Rotative Bearbeitung (Stanzen, Prägen, Drucken)
- Vermessung von Werkstücken
- Positions-Latch auf virtuellen oder realen Achsen

Eigenschaften:

- 2 digitale Eingänge
- bis zu 15 Time-Stamp-Einträge erfassbar
- geeignet für Schalter und Näherungsschalter

2 digitale Ausgänge mit Time Stamp-Funktion - EPM–S310

Dieser "schnelle Ausgang" erfasst bis zu zwei binäre Steuersignale vom übergeordneten Bussystem und überträgt diese mittels Time Stamp-Funktionalität zur Prozessebene. Der interne 16-Bit-Zähler (µs–Ticker) ist hier also die Basis zum Setzen eines bestimmten Ausgangs.

Schnellen Ausgänge werden eingesetzt für:

- Nockenschaltwerke, bei denen die Ausgänge sehr genau positionsabhängig gesetzt werden müssen.

Eigenschaften:

- 2 digitale Ausgänge
- FIFO-Speicher für bis zu 15 Time-Stamp-Einträge
- Ansteuerung über Prozessabbild

Hinweis: Da beide I/O-Komplettmodule einen eigenen 16-Bit-Zähler (µs–Ticker) haben, müssen diese durch den RWB-Controller synchronisiert werden.

Funktionsbaustein L_SMC_TouchProbe_FastIO

Funktionsbibliothek: L_SM3_DriveUtil.lib

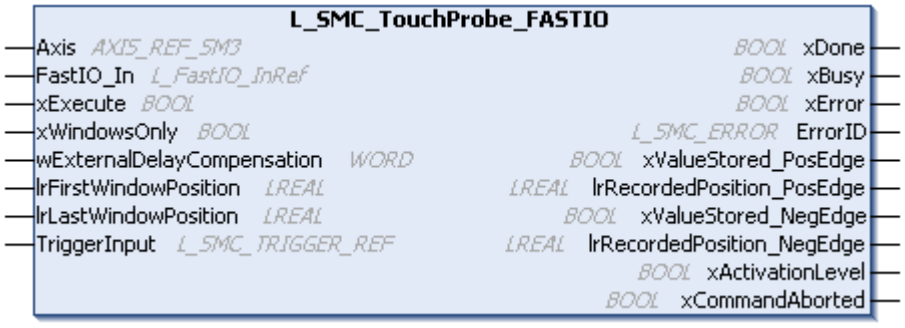
Ein schneller Eingangs des I/O-Komplettmoduls EPM-S207 liefert einen Time-Stamp-Eintrag (Latch des internen µs–Tickers), welcher in diesem Funktionsbaustein in eine Position umgerechnet wird.

Hinweis: Dieser Baustein orientiert sich am bereits vorhandenen TouchProbe-Baustein für Servo Drives 9400 und ECS-Geräte. Daher wird auf eine ausführliche Beschreibung an dieser Stelle verzichtet.

Beachten Sie:

- Die Laufzeiten der Achspositionen über den jeweiligen Feldbus werden im Funktionsbaustein L_SMC_TouchProbe_FastIO nicht kompensiert. Die Kompensation kann aber manuell über den Eingang wExternalDelayKompensation vorgegeben werden.
- Geber, die an einem Zähler-I/O-Komplettmodul (EMP-S600 bis EPM-S604) angeschlossenen sind, werden automatisch kompensiert. Auch die Zähler-Module verfügen über die Time-Stamp-Funktionalität, welche zusammen mit dem Positionswerten an die Steuerung übergeben werden.
- Für sehr genaue Touch-Probe-Messergebnisse empfehlen wir die Positionsmessung über einen Geber an einem Zähler-I/O-Komplettmodul zu realisieren.
- Die Anbindung eines Zähler-I/O-Komplettmoduls als SoftMotion-Achse finden Sie in der Beschreibung zum Funktionsbaustein L_SMC_FreeEncoder.

Hinweis: Wenn mehrere I/O-Komplettmodule mit Time-Stamp-Funktionalität (µs–Ticker) eingesetzt werden, müssen diese durch den RWB-Controller synchronisiert werden.



Ein-/Ausgänge (VAR_IN_OUT)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Axis	Referenz auf die FreeEncoder-Achse.
FastIO_InRef	Referenz I/O-Komplettmodul; dieser Baustein wird im Gerätebaum als Unterknoten automatisch hinzugefügt und instanziert, wenn z. B. das Modul EPM-S207 in den Gerätebaum eingefügt wird. Die Instanz dieses Bausteins muss dem Touch-Probe-Baustein übergeben werden.

Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
xExecute	Mit Setzen einer neuen positiven Flanke wird eine neue Messung gestartet. Alle Ausgänge des Bausteins werden auf die Initialwerte zurückgesetzt.
xWindowsOnly	Fensterauswertung aktivieren: Alle Messwerte außerhalb des definierten Fensters werden ausgeblendet.
wExternalDelayCompensation	Manuelle Kompensation der Laufzeiten der Achspositionen über den jeweiligen Feldbus. Die Eingabe der Verzögerungszeit erfolgt in [µs].
lrFirstWindowPosition	Untere Grenze des Fensters in SoftMotion-Einheiten [SM-Einheiten]
lrLastWindowPosition	Obere Grenze des Fensters in SoftMotion-Einheiten [SM-Einheiten]
TriggerInput	Siehe nachfolgende Beschreibung: L_SMC_Trigger_REF

TriggerInput / L_SMC_Trigger_REF:

Variable	Typ	Initialwert	Beschreibung
TouchProbeSelect	BYTE	1	Auswahl TP1 oder TP2 1: TP_Kanal1 -> DigIn 1 2: TP_Kanal2 -> DigIn 2
TouchProbeMode	BYTE	1	Auswahl Betriebsart 1: Einzelmessung; Execute-Flanke aktiviert die Funktion bis Done = 1 (TRUE) oder Abbruch durch FB L_SMC_AbortTrigger. 2: Kontinuierliche Messung; positive Execute-Flanke aktiviert die Funktion bis negative Execute-Flanke (Execute = FALSE) oder Abbruch durch FB L_SMC_AbortTrigger.
TouchProbeSignal	BYTE	1	Auswahl Nullimpuls oder Digitaler Eingang 1: Digitaler Eingang
TouchProbeEdge	BYTE	1	Konfiguration der Flankenauswertung 1: Triggern mit positiver Flanke 2: Triggern mit negativer Flanke 3: Triggern mit positiver oder negativer Flanke (Logik im FB)

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
xDone	Dieser Ausgang wird TRUE, sobald das Schreiben des Parameters fehlerfrei beendet ist.
xBusy	Dieser Ausgang ist TRUE, solange das Schreiben noch nicht beendet ist.
xError	Dieser Ausgang wird TRUE, wenn beim Schreiben ein Fehler aufgetreten ist. eError gibt die Fehlerursache aus.
ErrorID	Fehlernummer
xValueStored_PosEdge	Messwert gespeichert nach Triggern auf positive Flanke
lrRecordedPosition_PosEdge	Messwert beim Triggern mit der positiven Flanke in SoftMotion-Einheiten [SM-Einheiten], d. h. der Messwert wird in der selben Einheit an die PLC-Applikation geliefert, in der die Achse auch programmiert wird.
xValueStored_NewEdge	Messwert gespeichert nach Triggern auf negative Flanke.
lrRecordedPosition_NegEdge	Messwert beim Triggern mit der negativen Flanke in SoftMotion-Einheiten [SM-Einheiten], d. h. der Messwert wird in der selben Einheit an die PLC-Applikation geliefert, in der die Achse auch programmiert wird.
xActivationLevel	Pegel des TP-Eingangs zum Zeitpunkt der TP-Funktionsaktivierung im Antrieb. Die Applikation kann durch Auswertung des ActivationLevel bei der Beauftragung "positive oder negative Flanke" die Art der ersten aktiven Flanke erkennen.
xCommandAborted	Baustein abgebrochen über L_SMCAbortTrigger. Die Ausgänge "Busy" und "Done" werden zurückgesetzt.

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlermeldung	Beschreibung
411	TouchProbeSelect fehlerhaft: TP-Kanal nicht vorhanden TouchProbeSelect > 1 bei ECSxM TouchProbeSelect > 2 bei 9400
412	TouchProbeMode fehlerhaft: Nur die Werte 1 oder 2 sind zulässig.
413	TouchProbeEdge fehlerhaft:Nur die Werte 1, 2 oder 3 sind zulässig.
414	TouchProbeEdge fehlerhaft: Angeforderte Flankenauswertung nicht vorhanden TouchProbeEdge > 2 bei ECSxM (Nur positive oder negative Flanke möglich.) TouchProbeEdge > 4 bei 9400 (Art der Flankenauswertung wird nicht unterstützt.)
415	TP-Kanal nicht frei: TP-Kanal wurde bereits über anderen Baustein aktiviert.
416	Touch Probe ist bei aktuell angewähltem Prozessdaten-Mapping nicht möglich.
417	Interner Fehler / TP-Kanal nicht frei: TP-Kanal wurde bereits über anderen Baustein aktiviert.
418	Interner Fehler: Timeout bei der Beauftragung des Antriebes
419	Interner Fehler: Kommunikationsfehler
421	TouchProbeSignal fehlerhaft: Nur Wert 1 zulässig.
430	Die am Eingang Axis angegebene FreeEncoder-Achse existiert nicht
431	Die am Eingang Axis angegebene FreeEncoder-Achse ist fehlerhaft konfiguriert
432	TouchProbeEdge-Auswahl fehlerhaft
433	TouchProbeEdge Auswahl fehlerhaft

Funktionsbaustein L_SMC_AbortTrigger_FastIO

Funktionsbibliothek: L_SM3_DriveUtil.lib

Dieser Baustein dient zum Abbrechen eines laufenden Touchprobe-Auftrages am Baustein L_SMC_TouchProbe_FastIO.

Ein-/Ausgänge (VAR_IN_OUT)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Axis / AXIS_REF	Auswahl des Antriebes
FastIO_In / FastIO_InRef	Referenz I/O-Komplettmodul; dieser Baustein wird im Gerätebaum als Unterknoten automatisch hinzugefügt und instanziert, wenn z. B. das Modul EPM-S207 in den Gerätebaum eingefügt wird. Die Instanz dieses Bausteins muss dem Touch-Probe-Baustein übergeben werden.

Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Execute / BOOL	Abbrechen des laufenden Touchprobe-Auftrages. Mit dem Setzen einer neuen positiven Flanke werden alle Ausgänge des Bausteins auf die Initialwerte zurückgesetzt.
TriggerInput / L_SMC_Trigger_REF	Struktur zur Konfiguration der Messung
Axis / AXIS_REF	Auswahl des Antriebes

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
Done / BOOL	Funktionsbaustein L_SMC_TouchProbe wurde abgebrochen.
Error / BOOL	Fehlerkennung: Ein Fehler ist in der Beauftragung aufgetreten.
ErrorID / INT	Fehlernummer

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlernummer [hex]	Beschreibung
0x0001	Fehler in Beauftragung des Bausteins: Eingangsdaten stimmen nicht mit aktivierter TP-Beauftragung auf diesem Kanal überein. Ist auf dem TP-Kanal kein Auftrag aktiv, wird keine Fehlermeldung erzeugt.

Funktionsbaustein L_SMC_FastOutput

Funktionsbibliothek: L_SM3_DriveUtil.lib

Dieser Funktionsbaustein berechnet aus einer Position, die am Eingang fSwitchPosition vorgegeben wird, den entsprechenden Zeitpunkt (Latch des internen µs−Tickers), an dem die Ausgänge des I/O-Komplettmoduls EPM-S310 gesetzt werden. Da die Position in SoftMotion-Einheiten übergeben wird, wird als Referenz eine AxisRef übergeben.

Beachten Sie:

- Die Laufzeiten der Achspositionen über den jeweiligen Feldbus werden im Funktionsbaustein L_SMC_FastOutput nicht kompensiert. Die Kompensation kann aber manuell über den Eingang wExternalDelayKompensationD0/D1vorgegeben werden.
- Geber, die an einem Zähler-I/O-Komplettmodul (EMP-S600 bis EPM-S604) angeschlossenen sind, werden automatisch kompensiert. Auch die Zähler-Module verfügen über die Time-Stamp-Funktionalität, welche zusammen mit dem Positionswerten an die Steuerung übergeben werden.
- Damit der Ausgang in Abhängigkeit von der Position sehr genau gesetzt wird, empfehlen wir, die Positionsmessung über einen Geber an einem Zähler-I/O-Komplettmodul zu realisieren.
- Die Anbindung eines Zähler-I/O-Komplettmoduls als SoftMotion-Achse finden Sie in der Beschreibung zum Funktionsbaustein L_SMC_FreeEncoder.

Bitte beachten Sie: Sollen mehrere Schaltvorgänge nacheinander ausgeführt werden, so muss die Beauftragung in derselben Reihenfolge erfolgen, wie die Schaltvorgänge.

Beispiel: FreeEncoder-Achse dreht in positiver Richtung, d.h. Positionswerte wachsen.

1. Schaltvorgang soll erfolgen bei 60 Grad
2. Schaltvorgang soll erfolgen bei 90 Grad
3. Schaltvorgang soll erfolgen bei 120 Grad

Die Beauftragung muss in der Reihenfolge "1. Schaltvorgang", "2. Schaltvorgang" und abschießend "3. Schaltvorgang" erfolgen.

Ein-/Ausgänge (VAR_IN_OUT)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
Axis	Referenz auf die Achse. Es wird nur eine L_SMC_FreeEncoder-Achse unterstützt.
FastIO_Out	Referenz auf das I/O-Komplettmodul EPM-S310.

Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
xExecute	Mit Setzen einer neuen positiven Flanke wird eine neue Messung gestartet. Alle Ausgänge des Bausteins werden auf die Initialwerte zurückgesetzt.
byDigitalOutputSel	Auswahl des Ausganges.
xOutputState	Wert auf den der Ausgang geschrieben werden soll.
lrSwitchPosition	Positionswert, an dem der Ausgang gesetzt wird. Da die Position in SoftMotion-Einheiten übergeben wird, wird als Referenz eine AxisRef übergeben.
wExternalDelayCompensation	Die Laufzeiten der Achspositionen über den jeweiligen Feldbus werden nicht automatisch kompensiert. Für den D0-Ausgang erfolgt die manuelle Kompensation über diesen Eingang. Die Eingabe der Verzögerungszeit erfolgt in [µs].

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
xDone	Dieser Ausgang wird TRUE sobald das Schreiben des Parameters fehlerfrei beendet ist.
xBusy	Dieser Ausgang ist TRUE solange das Schreiben noch nicht beendet ist.
xError	Dieser Ausgang wird TRUE, wenn beim Schreiben ein Fehler aufgetreten ist. eError gibt die Fehlerursache aus.
ErrorID	Fehlernummer

Fehlermeldungen

Folgende Fehlermeldungen sind hinterlegt:

Fehlermeldung	Beschreibung
431	Die Konfiguration der am Eingang "Axis" angegebenen Achse ist fehlerhaft. Prüfen Sie die eingestellte Auflösung der Gebers.
416	Systemfehler. Bitte wenden Sie sich an den Lenze Support.
430	Die am Eingang "Axis"' angegebene Achse existiert nicht.
440	Auf der eingestellten Position ist ein Setzen des Ausganges auf Grund der Laufzeiten im System möglich.
441	Auf der eingestellten Position ist ein Setzen des Ausganges nicht möglich, da diese Position erst in mehr als 65 ms erreicht wird.
442	Fehlerhafte Auswahl des Ausgangs. Es stehen nur die Ausgänge 1 und 2 zur Verfügung.
443	Funktion kann nicht ausgeführt werden, da die Drehzahl der FreeEncoder-Achse zu niedrig ist.
444	FIFO-Überlauf beim I/O-Komplettmodul EPM-S310.

Funktionsbaustein L_SMC_FreeEncoder

Funktionsbibliothek: L_SM3_DriveUtil.lib

Der Funktionsbaustein L_SMC_TouchProbe_FastIO benötigt als Anschluss einen AxisRef-Baustein, der die Position und den Drehzahlwert eines Gebers liefert, um hieraus die Touch-Probe-Position zu ermitteln.

In Analogie zum "SMC_FreeEncoder-Device" mit einem untergeordneten AxisRef-Knoten ist es möglich, den existierenden Counter- und SSI-I/O-Modulen einen eigenen SoftMotion-Knoten manuell hinzuzufügen: L_SMC_FreeEncoder.

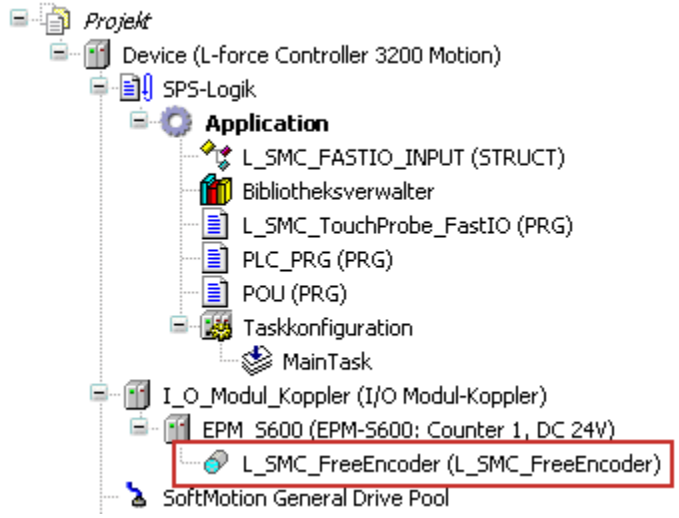


Abb. 1 Abbildung: Counter-I/O Modul mit AxisRef-Knoten

Dieser AxisRef-Baustein liefert neben den Prozesswerten Position, Drehzahl und Beschleunigung auch die Rohdaten des Counter- oder SSI-I/O-Moduls (also den „TimerTick“ zur gelesenen Position und die absolute Position).

Die Position von AxisRef wird hierbei derart kompensiert, dass sie sich auf den „Timer 0 Interrupt“ der Steuerung bezieht, der somit den Bezugspunkt bildet.

Zum „L_SMC_FreeEncoder-Device“ gehört ebenfalls eine spezielle Reiterkarte, die sich an der Reiterkarte des SM_Drive_Logical orientiert. Hier kann neben der Einstellung, ob ein lineares oder ein rotatives Maßsystem verwendet werden soll, auch eine Filterung von Position und Geschwindigkeit erfolgen.

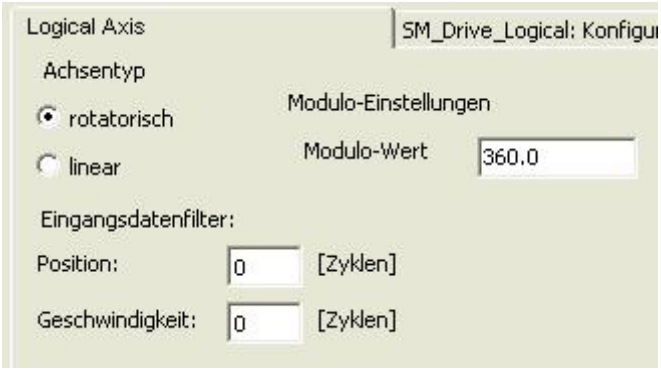
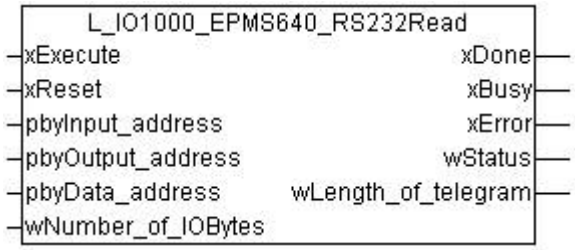


Abb. 2 Abbildung: Reiterkarte zum L_SMC_FreeEncoder

Funktionsbaustein L_IO1000_EPMS640_RS232Read

Funktionsbibliothek: L_SM3_DriveUtil.lib

Dieser Funktionsbaustein koordiniert den Datenempfang des I/O-Komplettmoduls EPM-S640 über RS232.



Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
xExecute / BOOL	Eine positive Flanke am Eingang löst einen Auftrag aus, d .h. sofern Daten vom I/O-Komplettmodul EPM-S64l empfangen wurden, werden diese eingelesen. Ein Rücksetzen des Eingangs während der Bearbeitung beeinflusst den Auftrag nicht. Mit xExecute =FALS“ und xReset=FALS“ wird der Baustein nach Abarbei-tung des aktuellen Auftrages zurückgesetzt und für einen neuen Reset / Datenauftrag vorbereitet.
xReset / BOOL	Eine positive Flanke am Eingang löst einen Reset des I/O-Komplettmoduls EPM-S640 aus, der das Modul neu initialisiert. Ein Rücksetzen des Eingangs während der Bearbeitung beeinflusst den Reset nicht. Mit xReset=FALSE und xExecute=FALSE wird der Baustein nach Abarbei-tung des aktuellen Auftrages zurückgesetzt und für einen neuen Reset / Datenauftrag vorbereitet. Hinweise: Vor dem ersten Datenempfang ist vom Anwender ein Reset durchzuführen! Der Reset ist ein Synchron-Reset, d .h. bei einer Durchführung am Read-Baustein wird das I/O-Komplettmodul EPM-S640 sowohl für Datenempfang als auch für Datensenden neu initialisiert. Wird gleichzeitig ein Reset und ein Datenauftrag angestoßen, wird der Reset auf Grund der höheren Priorität ausgeführt. Ein Reset kann auch während eines aktiven Auftrages durchgeführt werden und übernimmt in diesem Fall die Funktion eines Auftragsabbruchs.
pbyInput_address / POINTER_TO_BYTE	Anfangsadresse des Eingangsbereichs vom I/O-Komplettmodul EPM-S640 im Prozessabbild. Die Adresse muss mit dem ADR-Operator übergeben werden. Beispiel :"... pbyInput_address:=ADR(%IB4800) , ..."
pbyOutput_address / POINTER_TO_BYTE	Anfangsadresse des Ausgangsbereichs vom I/O-Komplettmodul EPM-S640 im Prozessabbild. Die Adresse muss mit dem ADR-Operator übergeben werden. Beispiel: "... pbyOutput_address:=ADR(%QB4800) , ..."

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
pbyData_address / POINTER_TO_BYTE	Anfangsadresse des Datenbereichs für die Empfangsdaten. Die Adresse muss mit dem ADR-Operator übergeben werden. Beispiel: "... pbyData_address:=ADR(%MB1024) , ..."
wNumber_of_IOBytes / WORD	Anzahl der verfügbaren Bytes im Prozessabbild. Die gültigen Werte hängen vom verwendeten Bussystem und der Modulkonfiguration ab: CAN: 8 Byte Profibus: 8, 16, 32, 60 Byte EtherCat :60 Byte

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
xDone / BOOL	Das Bit ist gesetzt, wenn ein Empfangsauftrag oder ein Reset fehlerfrei durchgeführt wurde.
xBusy / BOOL	Das Bit ist gesetzt, so lange ein Auftrag (Empfangen/Reset) bearbeitet wird. Nach Abschluss der Bearbeitung oder im Fehlerfall wird das Bit zurückgesetzt und xDone bzw. xError gesetzt.
xError / BOOL	Das Bit wird gesetzt, wenn ein Fehler vorliegt. Genauere Informationen zum Fehler liefert „wStatus“
wStatus / WORD	Kein Fehler (xError nicht gesetzt): 0x0000: kein Auftrag in Bearbeitung bzw. Auftrag fehlerfrei ausgeführt 0x8181: Sendeauftrag wird bearbeitet Fehler („xError“ gesetzt): 0x0202: ungültiger Wert für wNumber_of_IOBytes 0x0301: keine Adressen übergeben (pbyInput_address, pbyOutput_address, pbyData_address) 0x0407: Fehler beim Empfang
wLength_of_telegram / WORD	Länge des zuletzt empfangenen Telegramms. Das RS232-Modul kann Telegramme mit Längen zwischen 1 und 1024 Byte empfangen.

Hinweis: Pointer auf falsche Adresse oder auf Datenbereiche mit unzureichender Länge können Programmfehler oder gar einen Absturz der SPS zur Folge haben!

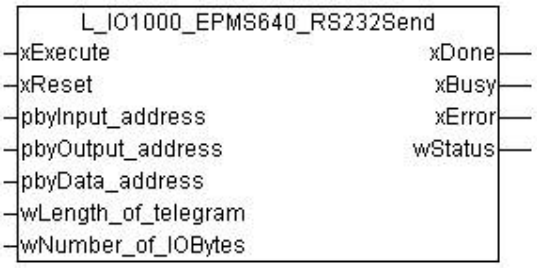
Beispiel

```
FUNCTION_BLOCK L_IO1000_EPMS640_RS232Read
VAR_INPUT
    xExecute: BOOL; (*Auftragsanstoß bei positiver Flanke /Freigabe Daten vom RS232 Modul lesen*)
    xReset: BOOL; (*Reset auslösen /Baustein zurücksetzen*)
    pbyInput_address: POINTER TO BYTE; (*Anfangsadresse des Eingangsbereichs im Prozessabbild*)
    pbyOutput_address: POINTER TO BYTE; (*Anfangsadresse des Ausgangsbereichs im Prozessabbild*)
    pbyData_address: POINTER TO  BYTE; (*Anfangsadresse des zu übertragenden Datenbereichs*)
    wNumber_of_IOBytes: WORD; (*Anzahl der verfügbaren Bytes im Prozessabbild 8/ 20/ 60*)
END_VAR
VAR_OUTPUT
    xDone: BOOL; (*Auftrag fertig ohne Fehler, Daten gesendet Parameter STATUS = 00h*)
    xBusy: BOOL; (*Information Bearbeitungsstatus*)
    xError: BOOL; (*Auftrag fertig mit Fehler, Parameter STATUS enthält die Fehlerinformation*)
    wStatus: WORD; (*Spezifikation des Fehlers bei ERROR = 1 /Information Bearbeitungsstatus, Auftrag wird abgearbeitet 0x8181h*)
    wLength_of_telegram: WORD;  (*Anzahl der zu übertragenden Bytes. 1 <= LEN <= 1024*)
END_VAR
```

Funktionsbaustein L_IO1000_EPMS640_RS232Write

Funktionsbibliothek: L_SM3_DriveUtil.lib

Dieser Funktionsbaustein koordiniert den Datenversand des I/O-Komplettmoduls EPM-S640 über RS232.



Eingänge (VAR_IN)

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
xExecute / BOOL	Eine positive Flanke am Eingang löst einen Auftrag aus, d. h. die definierten Daten werden vom I/O-Komplettmodul EPM-S640 versendet. Ein Rücksetzen des Eingangs während der Bearbeitung beeinflusst den Auftrag nicht. Mit xExecute =FALS“ und xReset=FALS“ wird der Baustein nach Abarbei-tung des aktuellen Auftrages zurückgesetzt und für einen neuen Reset / Datenauftrag vorbereitet.
xReset / BOOL	Eine positive Flanke am Eingang löst einen Reset des I/O-Komplettmoduls EPM-S640 aus, der das Modul neu initialisiert. Ein Rücksetzen des Eingangs während der Bearbeitung beeinflusst den Reset nicht. Mit xReset=FALSE und xExecute=FALSE wird der Baustein nach Abarbei-tung des aktuellen Auftrages zurückgesetzt und für einen neuen Reset / Datenauftrag vorbereitet. Hinweise: Vor der ersten Datensendung ist vom Anwender ein Reset durchzuführen! Der Reset ist ein Synchron-Reset, d .h. bei einer Durchführung am Write-Baustein wird das I/O-Komplettmodul EPM-S640 sowohl für Datensenden als auch für Datenempfang neu initialisiert. Wird gleichzeitig ein Reset und ein Datenauftrag angestoßen, wird der Reset auf Grund der höheren Priorität ausgeführt. Ein Reset kann auch während eines aktiven Auftrages durchgeführt werden und übernimmt in diesem Fall die Funktion eines Auftragsabbruchs.
pbyInput_address / POINTER_TO_BYTE	Anfangsadresse des Eingangsbereichs vom I/O-Komplettmodul EPM-S640 im Prozessabbild. Die Adresse muss mit dem ADR-Operator übergeben werden. Beispiel :"... pbyInput_address:=ADR(%IB4800) , ..."

Bezeichner / Datentyp	Info / Einstellmöglichkeiten
pbyOutput_address / POINTER_TO_BYTE	Anfangsadresse des Ausgangsbereichs vom I/O-Komplettmodul EPM-S640 im Prozessabbild. Die Adresse muss mit dem ADR-Operator übergeben werden. Beispiel: "... pbyOutput_address:=ADR(%QB4800) , ..."
pbyData_address / POINTER_TO_BYTE	Anfangsadresse des Datenbereichs für die Sendedaten. Die Adresse muss mit dem ADR-Operator übergeben werden. Beispiel: "... pbyData_address:=ADR(%MB0) , ..."
wLength_of_telegram / WORD	Anzahl der zu versendenden Bytes ab der Anfangsadresse "pbyData_address". Gültig sind Längen zwischen 1 und 1024 Byte.
wNumber_of_IOBytes / WORD	Anzahl der verfügbaren Bytes im Prozessabbild. Die gültigen Werte hängen vom verwendeten Bussystem und der Modulkonfiguration ab: CAN: 8 Byte Profibus: 8, 16, 32, 60 Byte EtherCat: 60 Byte

Ausgänge (VAR_OUT)

Bezeichne / Datentyp	Wert / Bedeutung
xDone / BOOL	Das Bit ist gesetzt, wenn ein Sendeauftrag oder ein Reset fehlerfrei durchgeführt wurde.
xBusy / BOOL	Das Bit ist gesetzt, so lange ein Auftrag (Senden/Reset) bearbeitet wird. Nach Abschluss der Bearbeitung oder im Fehlerfall wird das Bit zurückgesetzt und xDone bzw. xError gesetzt.
xError / BOOL	Das Bit wird gesetzt, wenn ein Fehler vorliegt. Genauere Informationen zum Fehler liefert „wStatus“
wStatus / WORD	Kein Fehler (xError nicht gesetzt): 0x0000: kein Auftrag in Bearbeitung bzw. Auftrag fehlerfrei ausgeführt 0x8181: Sendeauftrag wird bearbeitet Fehler („xError“ gesetzt): 0x0202: ungültiger Wert für wNumber_of_IOBytes 0x0517: ungültiger Wert für wLength_of_telegram 0x070A: Übertragung fehlgeschlagen 0x0301: keine Adressen übergeben (pbyInput_address, pbyOutput_address, pbyData_address)

Hinweis: Pointer auf falsche Adresse oder auf Datenbereiche mit unzureichender Länge können Programmfehler oder gar einen Absturz der SPS zur Folge haben!

Beispiel

```
FUNCTION_BLOCK L_IO1000_EPMS640_RS232Send
VAR_INPUT
    xExecute: BOOL; (*Auftragsanstoß bei positiver Flanke /Freigabe Daten an RS232 Modul senden*)
    xReset: BOOL; (*Reset auslösen /Baustein zurücksetzen*)
    pbyInput_address: POINTER TO BYTE; (*Anfangsadresse des Eingangsbereichs im Prozessabbild*)
    pbyOutput_address: POINTER TO BYTE; (*Anfangsadresse des Ausgangsbereichs im Prozessabbild*)
    pbyData_address: POINTER TO  BYTE; (*Anfangsadresse des zu übertragenden Datenbereichs*)
    wLength_of_telegram: WORD;  (*Anzahl der zu übertragenden Bytes. 1 <= LEN <= 1024*)
    wNumber_of_IOBytes: WORD; (*Anzahl der verfügbaren Bytes im Prozessabbild 8/ 20/ 60*)
END_VAR
VAR_OUTPUT
    xDone: BOOL; (*Auftrag fertig ohne Fehler, Daten gesendet Parameter STATUS = 00h*)
    xBusy: BOOL; (*Information Bearbeitungsstatus*)
    xError: BOOL; (*Auftrag fertig mit Fehler, Parameter STATUS enthält die Fehlerinformation*)
    wStatus: WORD; (*Spezifikation des Fehlers bei ERROR = 1 /Information Bearbeitungsstatus, Auftrag wird abgearbeitet 0x8181h*)
END_VAR
```

2.2 CANopen / PROFIBUS / EtherCAT

In den Kommunikationshandbüchern "Steuerungstechnik CANopen" und "Steuerungstechnik EtherCAT" finden Sie ausführliche Informationen zur Inbetriebnahme und Diagnose von "Logic"- und "Motion"-Feldgeräten.

Hinweis: PROFIBUS kann in der Lenze Steuerungstechnik nur als "Logic-Bus" verwendet werden.

Sonderfall "Querkommunikation"

Eine Querkommunikation ist nur bei Logic-Geräten möglich. Hierzu muss das erforderliche PDO-Mapping im »PLC Designer« angelegt werden. Beim Starten der PLC wird die komplette Konfiguration/PDO-Mapping in die Geräte geschrieben. Die Einträge aus dem »Engineer« werden dabei überschrieben.

Hinweis: Das PDO-Mapping bei Achsmodulen ECSxM ist statisch und kann daher nicht verändert werden.

2.3 PLC-Programm zur Dokumentation einer Zykluszeit-Verletzung

Eine Zykluszeit-Verletzungen können Sie im Logbuch des L-force Controllers mittels nachfolgend beschriebenem Code dokumentieren.

Bei der Ermittlung der benötigten Zykluszeit eines PLC-Projektes, beachten Sie bitte die in den Kommunikationshandbüchern "Steuerungstechnik CANopen" und "Steuerungstechnik EtherCAT" beschriebenen Verfahren (Abschnitt "Minimale Zykluszeit des PLC-Projektes bestimmen").

Funktion:

- Mit Variablen vom Typ "Task_Info" aus der Systembibliothek "CmpIecTask" können Informationen über eine Task abgefragt werden.
- Mit der Funktion "L_AddLog" kann aus der PLC ein Logeintrag im Logbuch des L-force Controllers erzeugt werden.
- Falls der gemessene Jitter größer als die Zykluszeit ist, ist ein Task-Zyklus ausgefallen und es wird ein entsprechender Log-Eintrag im LogBook des L-force Controllers erzeugt.

Voraussetzung:

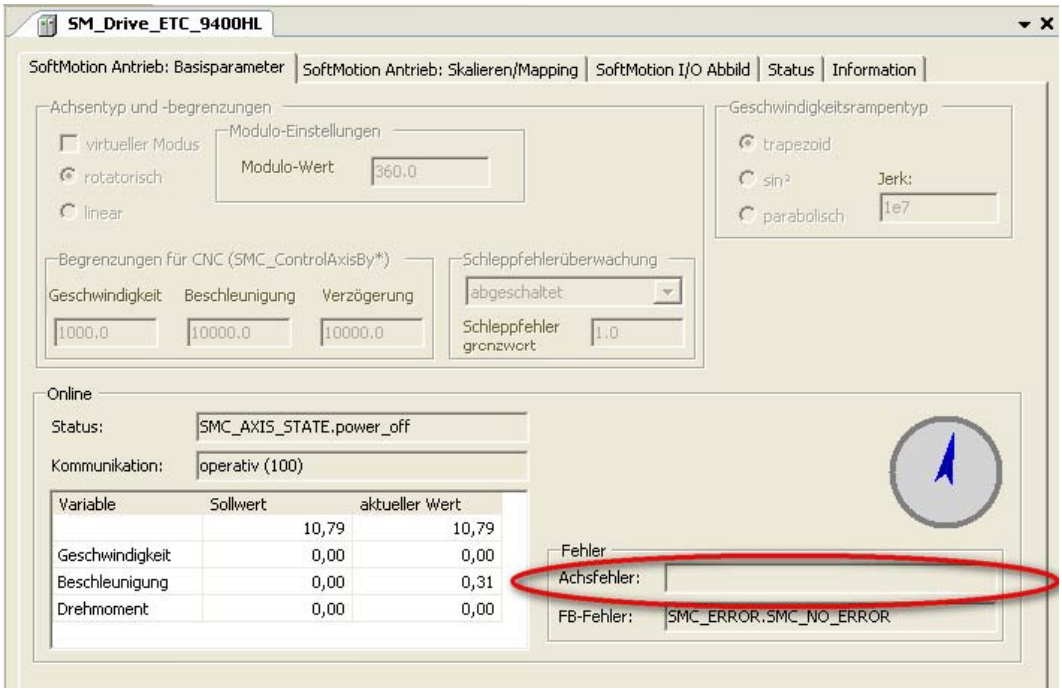
- Die Bibliotheken "CmpIecTask" und "SysTypes" sind im PLC Designer-Projekt eingebunden.
- Der nachfolgend genannte Code muss in der zu untersuchenden Task zyklisch aufgerufen werden.

Beispiel für die Überwachung der Zykluszeitüberwachung:

```
VAR
   pIecTaskResult : POINTER TO RTS_IEC_RESULT;
   tIecTaskHandle : RTS_IEC_HANDLE;
   pIecTaskInfo : POINTER TO Task_Info;
   tTaskInfo : Task_Info;
END_VAR
-------------------------------------------------------------------------
tIecTaskHandle := IecTaskGetCurrent( pIecTaskResult );
pIecTaskInfo := IecTaskGetInfo2( tIecTaskHandle, pIecTaskResult );
tTaskInfo := pIecTaskInfo^;
IF ABS(tTaskInfo.iJitter) > TIME_TO_DINT(tTaskInfo.tInterval) THEN
    L_AddLog(eSeverity:= L_TSeverity.L_LogWarning,
      eFacility:= L_TFacility.L_PLC_USER_LOG_SCOPE1,
      sLogPayLoad:= 'Verletzung der Zykluszeit PRG');
END_IF
```

2.4 Auf der SoftMotion-Reiterkarte gemeldete Achsfehler

Im Online Zustand des »PLC Designer« werden auf der SoftMotion-Reiterkarte "Basisparameter" evtl. vorhanden Achsfehler ausgegeben:



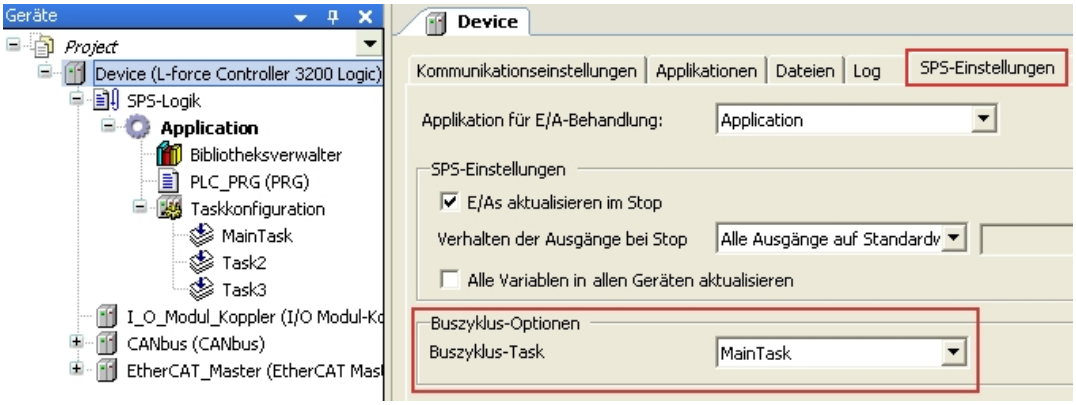
Nummer des Achsfehlers	Bedeutung
2000	SDO Zugriff: Timeout beim Schreiben
2001	SDO Zugriff: Timeout beim Lesen
2010	Fehlerhafte Zykluszeit der zugeordneten Motiontask
2011	Fehlerhafter ‚Maximal Drehmoment' Parameter im Antrieb
2020	Falsche Applikation auf dem Antrieb

2.5 Task zur Prozessabbild-Übernahme zuordnen

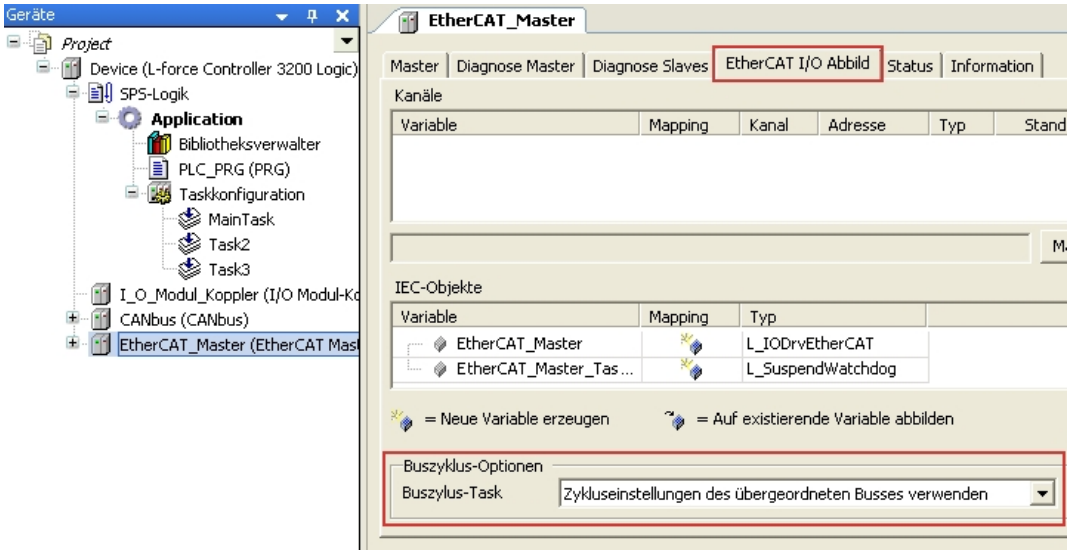
Die Übernahme des Prozessabbildes eines Gerätes lässt sich gezielt einer bestimmten Task zuordnen.

Dazu stehen die folgenden Eingabefelder zur Verfügung:

Steuerungsebene



Busmaster-Ebene (z. B. EtherCAT-Master)



Greifen in einem PLC-Projekt mehrere Tasks auf das Prozessabbild eines Bus-Masters zu, ist immer die schnellste Task als Buszyklus-Task einzutragen. Die Buszyklus-Tasks der verschiedenen Bussysteme (EtherCAT / CAN / PROFIBUS) können hierbei voneinander unabhängig eingestellt werden.

2.6 Datenkonsistenz beim Zugriff auf das Prozessabbild

In einem PLC-Projekt können mehrere Tasks auf das Prozessabbild eines Bus-Masters zugreifen. Abhängig von der konfigurierten Priorität und Zykluszeit, können sich Tasks hierbei gegenseitig unterbrechen.

Die Datenkonsistenz ist dabei vom System nur beim Zugriff auf Datentypen der folgenden Länge sichergestellt:

- Byte
- Word
- Dword

Ebenfalls abgesichert sind Bit-Zugriffe, da auch hier nicht unterbrechbare (d. h. atomare) Befehle zur Anwendung kommen.

Vom System nicht abgesichert sind dagegen Zugriffe auf Datenstrukturen (Records). In diesem Fall muss die Datenkonsistenz vom Anwender sichergestellt werden.

Dies wird da durch erreicht, dass nur jeweils eine Task auf eine bestimmte Datenstruktur im Prozessabbild schreibend zugreift.

3 Konzepte, Basiskomponenten

3.1 Einführung und Basiskonzepte

Der PLC Designer ist ein geräteunabhängiges Steuerungsprogrammiersystem.

Gemäß der Norm IEC 61131-3 unterstützt es alle Standard-Programmiersprachen, erlaubt aber auch das Einbinden von C-Routinen. In Kombination mit dem PLC Designer Laufzeitsystem ermöglicht es die Programmierung mehrerer Steuerungen innerhalb eines Projektes.

Beachten Sie die folgenden Basiskonzepte, die das Programmieren mit dem PLC Designer bestimmen: **Objektorientierung**: Der Gedanke der Objektorientierung äußert sich nicht nur durch die Verfügbarkeit von entsprechenden Programmierelementen und Programmfunktionen, sondern auch in der Struktur und der Versionsverwaltung vom PLC Designer und in der Projektorganisation. "**Multi-device**" und "**Multi-Application**" Gebrauch eines PLC Designer-Projekts ist möglich durch instanzierbare und gemeinsam nutzbare Programmeinheiten. Verschiedene Applikationen können auf einem Gerät betrieben werden. Parametrierbare und programmierbare Geräte können innerhalb desselben Projektes angesprochen werden.

- **Komponentenbasierte** Struktur des Programmiersystems: Welche Funktionalität das Programmiersystem dem Anwender bietet (Editoren, Menüs etc.) hängt von den gerade verwendeten Komponenten ab (plug-ins). Dies wiederum ist definiert im sogenannten Profil. Es gibt Systemkomponenten, die unverzichtbar sind, sowie optionale Komponenten. Zusätzlich zu den Standard Lenze-Plug-Ins können **kundenspezifische Plug-ins** verwendet werden, die mit Hilfe des **Automation Platform** Toolkit erstellt werden.
- **Versionsverwaltung**: Verschiedene Versionen der PLC Designer-Komponenten können parallel installiert sein und dann kann mit der jeweils gewünschten gearbeitet werden. Dies umfaßt auch den gerätespezifischen Gebrauch verschiedener Compiler-Versionen. Einzelne Funktionalitäten können hinzugefügt werden ohne einen Update der gesamten Version zu erfordern.
- **Projektorganisation**: Diese ist ebenfalls durch den Gedanken der Objektorientierung geprägt:
  Ein PLC Designer-Projekt enthält sowohl ein Steuerungsprogramm, zusammengesetzt aus verschiedenen Programmierobjekten, als auch die Definitionen der "Ressourcen" die benötigt werden, um Instanzen dieses Programms, als "Applikations"-Objekte gehandhabt, auf einem bestimmten Zielsystem (Gerät, Modul, Steuerung) zu betreiben.

Somit gibt es zwei Haupttypen von Objekten in einem Projekt:

Programmierobjekte (POUs):
Dies sind Programme, Funktionen, Funktionsblöcke, Methoden, Schnittstellen, Aktionen, Datentyp-Definitionen usw.
Programmierobjekte, die projektweit, d.h. für alle im Projekt definierten Applikationen, instanziert werden können sollen, müssen im POUs-Fenster verwaltet werden.
Das **Instanzieren** erfolgt dadurch, dass ein Programm-POU von der Task einer Applikation aufgerufen wird.
Programmierobjekte, die nur im Geräte-Fenster verwaltet werden, d.h. direkt einer Applikation zugeordnet sind, können nicht von anderen Applikationen instanziert werden.

Ressourcen-Objekte (Gerätebaum):
Dies sind Geräte-Objekte, Applikationen, Taskkonfigurationen, Rezepturverwalter etc., und werden NUR im Geräte-Fenster, also im "Gerätebaum", verwaltet. Für das Anlegen der Geräteobjekte im Gerätebaum gelten bestimmte Regeln, nach denen die zu steuernde Hardware-Umgebung abgebildet werden muss. Durch die hierarchische Anordnung von Applikations- und Geräte-Objekten werden die Gültigkeitsbereiche von Objekten wie Bibliotheken und GVLs etc. definiert.

- **Codegenerierung** durch integrierte Compiler und Verwendung von Maschinen-Code ermöglicht kurze Ausführungszeiten.
- **Datentransfer** zum Steuerungsgerät: Der Datentransfer zwischen PLC Designer und der Steuerung erfolgt über einen **Gateway** (Komponente) und ein **Laufzeitsystem**. Eine komplette Online-Funktionalität für das Überwachen des Programms auf der Steuerung ist verfügbar.
- Vordefinierte **Feature-Sets** dienen dazu, um zwischen einer "Standard" Benutzeroberfläche mit einem reduzierten Funktionalitätsumfang und weniger Komplexität und einer "Professional" Umgebung, die alle Funktionalitäten umfaßt, wählen zu können. Beim ersten Öffnen des Programmiersystems nach der Erstinstallation wird bereits dazu aufgefordert. Aber auch danach kann nochmals umgestellt werden und auch eine benutzerdefinierte Anpassung des aktuell verwendeten Feature Sets ist möglich. Zu den Unterschieden zwischen Standard und Professional Feature-Set siehe die Beschreibung des Feature Optionen.

3.2 Unterschiede zum PLC Designer V2.3

Grundsätzlich können Projekte, die mit PLC Designer V2.3 erstellt wurden, geöffnet und weiter bearbeitet werden.

PLC Designer V3.x bietet die folgenden Erweiterungen und Verbesserungen:

Objektorientierung beim Programmieren und in der Projektstruktur

- Erweiterungen für Funktionsblöcke (Funktionsbausteine): Eigenschaften, Schnittstellen, Methoden, Vererbung, Methodenaufruf
- Gerätegebundene Applikationen als Instanzen von unabhängigen Programmierobjekten

Neu bezüglich Datentypen

- ANY_TYPE
- UNION
- LTIME
- References
- Enumerationen: Basisdatentyp kann angegeben werden
- di : DINT := DINT#16#FFFFFFFF; nicht erlaubt

Neu bezüglich Operatoren und Variablen

- Neue Gültigkeitsbereich-Operatoren, erweiterte Namensräume
- Funktionszeiger ersetzen den INSTANCE_OF Operator
- Init Methode ersetzt den INI Operator
- Exit Methode
- Ausgangsvariablen in Funktions- und Methoden-Aufrufen
- VAR_TEMP, VAR_STAT
- Beliebige Ausdrücke für Variablen-Initialisierung
- Zuweisung als Ausdruck
- Index-Zugriff mit Zeigern und Strings

Neues Visualisierungskonzept

- Ein Visualisierungs-Editor steht bereit, der standardmäßig mit einer Werkzeugbox und einem Editor für die Elementeigenschaften zusammenarbeitet.
- Teile der Visualisierungs-Funktionalität sind gemäß IEC 61131 realisiert und werden somit über Bibliotheken bereit gestellt.
- Ein internes Laufzeitsystem führt die wichtigsten Visualisierungsfunktionen aus.
- Textlisten und Bildersammlungen

Neues Konzept für Benutzerverwaltung und Sicherheit

- Benutzerkonten, Benutzergruppen, gruppengebundene Rechte für Zugriff und Aktionen auf einzelne Objekte

Neues in Editoren

- ST-Editor: Klammerungen, Umbrüche, Code-Komplettierung, Inline Monitoring, Inline set/reset-assignment
- FUP, KOP und AWL sind ineinander konvertierbar und haben einen gemeinsamen Editor
- AWL-Editor als Tabelleneditor
- FUP/KOP/AWL-Editor: Der Hauptausgang in Bausteinen mit mehrfachen Ausgängen kann gesetzt werden
- FUP/KOP/AWL-Editor: Keine automatische Aktualisierung der Bausteinparameter
- FUP/KOP/AWL-Editor: Verzweigungen und "Netzwerke in Netzwerken"
- AS-Editor: Nur 1 Schritt-Typ, Makros, Mehrfachselektion von unabhängigen Elementen, keine Syntaxüberprüpfung während des Editierens

Bibliotheksverwaltung

- Mehrere Versionen einer Bibliothek im gleichen Projekt möglich; eindeutige Zugriffe durch Angabe des Namensraums
- Installation in Repositories, Automatische Aktualisierung, Debugging

und weiteres....

- konfigurierbare Menüs, Werkzeugleiste und Tastaturbedienung
- kundenspezifische Komponenten über Plug-Ins hinzufügbar
- Steuerungskonfiguration und Taskkonfiguration sind im Gerätebaum integriert
- Unicode-Unterstützung
- einzeilige Kommentare: // Kommentar
- CONTINUE in Schleifen
- Mehrfachselektion im Objektbaum
- Online-Hilfe in Benutzeroberfläche integriert
- Bedingte Kompilierung
- Bedingte Haltepunkte
- Debuggen: Ausführen bis Cursor, Ausführen bis Rücksprung
- Feldbus-Treiber gemäß IEC 61131-3
- Symbol- und Steuerungskonfiguration in Applikation
- Freie Speicherzuweisung für Code und Daten
- Jedes Objekt kann als "intern" oder "extern" (Spätes Binden im Laufzeitsystem) definiert werden
- Verbindung zu externen Datenquellen

Kompatibilität mit PLC Designer V2.3 Projekten

- Projekte anderen Formats, u.a. mit PLC Designer V2.3 erstellte Projekte, können geöffnet werden, wenn der entsprechende Konverter zur Verfügung steht. Siehe Befehl 'Projekt öffnen'. Dabei kann festgelegt werden, wie eingebundene Bibliotheken und Geräte gehandhabt werden sollen.
- Syntaktische und semantische Einschränkungen gegenüber PLC Designer V2.3 Projekten:
  - FUNCTIONBLOCK ist kein anstelle von FUNCTION_BLOCK gültiges Schlüsselwort mehr
  - TYPE (Strukturdeklaration) benötigt ein nachfolgendes ":"
  - ARRAY-Initialisierung verlangt eckige Klammern
  - Lokale Deklaration einer Enumeration ist nur mehr innerhalb von TYPE / END_TYPE möglich
  - INI nicht länger unterstützt (wird durch die Init Methode ersetzt)
  - In Funktionsaufrufen ist es nicht länger möglich, explizites mit impliziten Zuweisungen zu mischen; somit kann allerdings die Anordnung der Eingangsparameter verändert werden:
    fun(formal1 := actual1, actual2); // -> Fehlermeldung
    fun(formal2 := actual2, formal1 := actual1); // gleiche Semantik wie
    fun(formal1 := actual1, formal2 := actual2);
- Pragmas (Import von PLC Designer V2.3 Pragmas noch nicht implementiert)

3.3 Profile und Plug-Ins

Wenn "PLC Designer without Profile" über das Startmenü gewählt wird, wird man aufgefordert, ein "Profil" zu wählen. (Wenn direkt mit "PLC Designer V3.x" aus dem Startmenü oder über das Desktop-Symbol gestartet wird, wird automatisch das Standard-Profil verwendet.).

Ein Profil definiert den Umfang der in der Benutzeroberfläche verfügbaren Komponenten (**plug-in**s).

Jedes Plug-In ist mit einen Namen, einer Versionsnummer und Versionseinschränkungen (constraints) gekennzeichnet.

Neben den von Lenze zur Verfügung gestellten Komponenten können auch kundenspezifische Plug-Ins integriert sein.

Beispiele für essentielle Komponenten sind 'Online Manager' oder 'Runtime System'. Diese müssen auf jeden Fall im System enthalten sein. Andere von Lenze bereitgestellte Plug-Ins, wie verschiedenen Editoren, Code-Generatoren, Kommunikationstreiber etc. können optional verwendet werden.

Zum Erstellen von Plug-Ins für hersteller- bzw. kundenspezifische Komponenten gibt es ein Toolkit (**Automation Platform Toolkit**, siehe separate Dokumentation).

Für die Anzeige der Komponenten eines Profils und für Änderungen am Profil benötigt man den **Installation and Profile Manager** (IPM.exe). Er bietet einen Dialog zum Installieren von Plug-Ins auf dem lokalen System und einen weiteren Dialog für die Konfiguration von Profilen auf Basis der installierten Plug-Ins.

Profil-Konfigurationen werden in Dateien mit Erweiterung ".profile" gespeichert.

Beachten Sie bitte folgendes:

- Mehrere Versionen können gleichzeitig installiert sein.
- Unterschiedliche Versionen des Laufzeitsystems und Programmiersystems können zusammen verwendet werden.
- Jeder Service-Mitarbeiter hat auf seinem Rechner alle Versionen verfügbar.
- Jedes Projekt weiß, mit welcher Version es erstellt wurde. Wenn es in einer anderen Version geöffnet wird, wird der Anwender auf mögliche bzw. nötige Aktualisierungen bzgl. Speicherformat, Bibliotheksversionen etc. hingewiesen.
- Individuelle Funktionen können hinzugefügt werden, ohne dass eine Aktualisierung aller PLC Designer-Komponenten nötig wird.

3.4 Projekt

Ein Projekt enthält die POU Objekte aus denen sich ein **Steuerungsprogramm** zusammensetzt, wie auch die Definitionen der Ressourcen-Objekte, die benötigt werden, um eine oder mehrere Instanzen des Programms (Applikation) auf einem bestimmten Zielsystem (Steuerungen, Geräte) auszuführen. Projektweit verfügbare POU-Objekte werden im **POUs** Fenster verwaltet, gerätespezifische Ressourcen-Objekte und applikationsspezifische POU-Objekte im Geräte- Fenster.

Ein Projekt wird in einer Datei **<projectname>.project gespeichert.**

Projektspezifische Konfigurationen können in den Anpassen- und Optionen-Dialogen vorgenommen werden.

Hinweis: Das Aussehen und die Eigenschaften der Benutzeroberfläche werden mit dem PLC Designer, nicht mit dem Projekt gespeichert. Die in der Oberfläche verfügbaren Komponenten werden durch das Profil bestimmt.

Projekt-Vorlage

Die Basiskonfiguration eines neuen Projekts (Menüstruktur, vordefinierte Objekte) wird bestimmt durch die verwendete Projekt-**Vorlage**. Die Vorlage wird ausgewählt, wenn ein neues Projekt über den Befehl Neues Projekt angelegt wird.

Projekteinstellungen und Projektinformationen

Die Projekteinstellungen sind standardmäßig als Objekt im POUs Fenster vorhanden, die Projektinformationen werden automatisch als solches dort aufgenommen, sobald sie das erste Mal über den entsprechenden Menübefehl (standardmäßig im Projekt-Menü) geöffnet wurden.

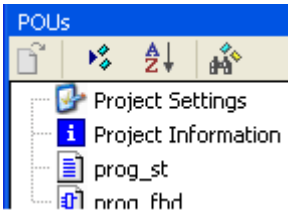


Abb. 3 POUs Fenster

In den Dialogen der Projekteinstellungen können verschiedene Einstellungen für das gerade bearbeitete Projekt vorgenommen werden, wie z. B. was geschehen soll, wenn ein Anwender das Projekt mit einer anderen Version des Entwicklungssystems öffnet.

In den Dialogen der Projektinformationen finden sich Angaben zum gerade bearbeiteten Projekt, wie z. B. Daten zur Datei, Statistiken zu den Objekten, Name des Autors etc.

Projekte verschmelzen bzw. kopieren

Ein Projekt bzw. einzelne Objekte eines Projekts können in ein anderes Projekt kopiert werden. Sehen Sie hierzu den Befehl Projekt kopieren.

PLC Designer V2.3 Projekte

Projekte, die mit Version 2.3 erstellt wurden, können geöffnet und in das V3.x Format konvertiert werden. Allerdings wird empfohlen, einige syntaktische Vorbereitungen zu treffen.

Wenn Bibliotheken oder Geräte in ein V2.3 Projekt eingebunden sind, kann vor dem Konvertieren des Projekts entschieden werden, ob sie weiterhin im Projekt verwendet oder durch andere ersetzt werden sollen, oder ob die Referenzen entfernt werden sollen.

3.5 Unterstützte Programmiersprachen

Alle in der IEC-Norm IEC 61131 aufgeführten Programmiersprachen werden mittels speziell angepasster Editoren unterstützt:

- FUP/KOP/AWL-Editor für Funktionspläne (FUP), Kontaktpläne (KOP) und Anweisungslisten (AWL)
- AS-Editor für Ablaufsprache (AS)
- ST-Editor für Strukturierten Text

Zusätzlich wird ein Editor zur Verfügung gestellt, der das Programmieren in der Sprache CFC (Continuos Function Chart) unterstützt:

- CFC-Editor für Continuous Function Chart.

3.6 Was ist ein Program Organization Unit Objekt "POU"

Lassen Sie sich bitte bezüglich des Begriffs "**POU"** nicht verwirren: Als POUs werden Programmiereinheiten (Objekte) bezeichnet, aus denen sich ein Steuerungsprogramm zusammensetzt und die zum Programm gehören.

POU = Program Organization Unit = Programmier-Organisationseinheit.

POUs, die im "**POUs**"-Fenster verwaltet werden, sind nicht gerätespezifisch, sondern projektweit verwendbar und können ggfs. für die Verwendung auf einem im Projekt beschriebenen Gerät in der gerätespezifischen Applikation "instanziert" werden. Zu diesem Zweck werden Programm-POUs über die eine Task der jeweiligen Applikation aufgerufen.

POUs, die NUR im Geräte-Fenster verwaltet werden, d. h. dort im Gerätebaum direkt unter einer Applikation eingefügt sind, können nur von Applikationen verwendet werden, die unterhalb dieser Applikation eingerückt bereitstehen ("Kind"-Applikationen). Siehe weitere Informationen bei der Beschreibung des Gerätebaums und des Applikations-Objekts.

Aber: "POU" wird im Objekt hinzufügen-Dialog auch als Name einer bestimmten **Unterkategorie** dieser Objekte verwendet und bezeichnet an dieser Stelle nur Programme, Funktionsbausteine und Funktionen!

Somit ist ein Program Organization Unit Objekt generell eine Programmiereinheit, ein Objekt, das nicht-gerätespezifisch im "POUs"-Fenster oder gerätespezifisch im "Geräte"-Fenster verwaltet wird und in einem Editorfenster angezeigt und bearbeitet werden kann.

Ein POU Objekt kann ein Programm, eine Funktion, ein Funktionsbaustein (Funktionsblock), oder auch eine Methode, Aktion, Interface, DUT (Data Unit Type, Datentyp) oder eine externe Datei beliebigen Formats sein.

Beachten Sie die Möglichkeit, bestimmte **Eigenschaften** (wie z. B. Vorgaben für den Übersetzungsvorgang etc.) für jedes POU Objekt getrennt zu definieren.

Um ein POU Objekt im Projekt einzufügen, verwenden Sie den Befehl 'Objekt hinzufügen'.

Die folgenden Typen von **POU Objekten** können verwendet werden:

- Aktion
- DUT (Datentyp)
- Externe Datei
- Globale Variablenliste
- Globale Network Variablenlistet
- Bildersammlung
- Schnittstelle
- Bibliotheksverwalter
- Methode
- Eigenschaft
- POU
- Textliste
- Visualisierung

Neben den POU-Objekten werden "Ressourcen" für das Ausführen des Programms auf dem Zielsystem benötigt (Applikation, Taskkonfiguration etc.). Diese werden im "Geräte"-Fenster im Gerätebaum verwaltet.

3.7 Programm-Organisationseinheiten - "POUs"

3.7.1 POU

Der Begriff "POU" auf dieser Ebene bezeichnet eine Program Organization Unit (Programm-Organisationseinheit) vom Typ Programm, Funktion oder Funktionsblock (Funktionsbaustein).

Um ein POU Objekt einzufügen, wählt man den passenden Eintrag im vorhandenen POUs- oder Gerätebaum (beispielsweise ein Applikationsobjekt), verwendet Befehl Objekt hinzufügen vom Kontextmenü und wählt "POU" aus dem erscheinenden Untermenü. Daraufhin öffnet sich der Dialog POU hinzufügen, in dem man die POU hinsichtlich Name, Typ und Implementierungssprache konfiguriert. Im Falle eines Funktionsbausteins können (wenn vom Profil untersützt) EXTENDS and IMPLEMENTS Eigenschaften hinzugefügt werden, im Falle einer Funktion muss auch der Rückgabewert angegeben werden.

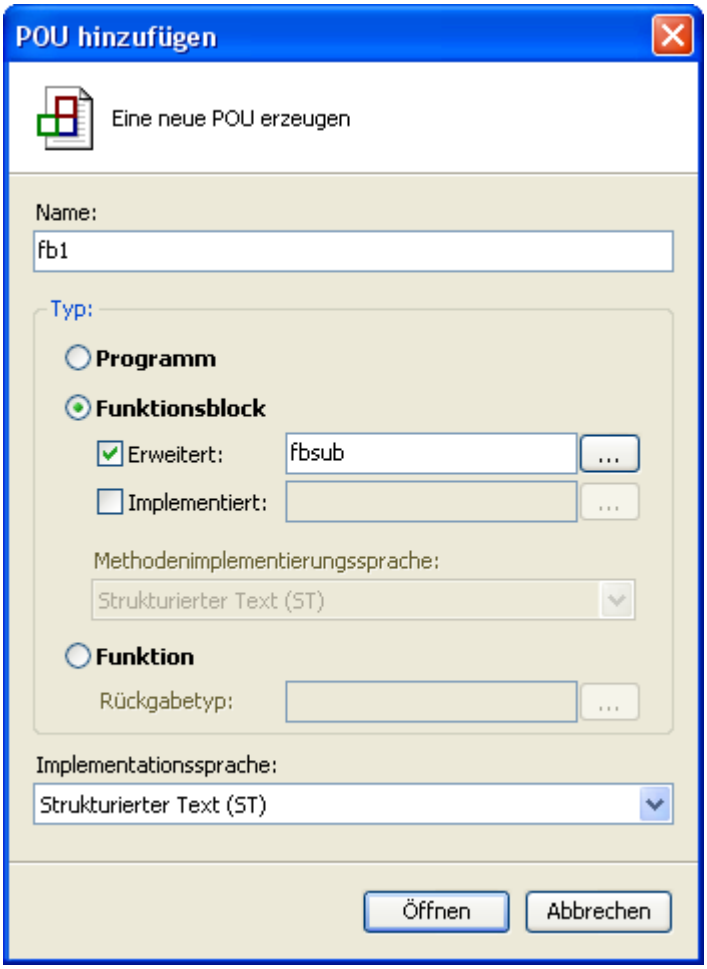


Abb. 4 POU hinzufügen Dialog, Beispiel

Je nach Typ kann die POU dann um Methoden, Eigenschaften, Aktionen, Transitionen erweitert werden; dies ebenfalls über den Befehl 'Objekt hinzufügen'.

Die hierarchische Abarbeitung der einzelnen POUs, die einer Applikation zugeordnet sind, hängt von der gerätespezifischen Konfiguration (Aufrufbaum) ab.

Jede POU besteht aus einem **Deklarationsteil** und einem **Implementationsteil**. Der Implementationsteil wird in einer der verfügbaren Programmiersprachen geschrieben, z.B. AWL, ST, AS, FUP, KOP oder CFC.

PLC Designer unterstützt alle in der Norm 61131-3 beschriebenen POUs. Wenn Sie diese Standard-POUs in ihrem Projekt, verwenden wollen, muss die Bibliothek standard.library eingebunden sein.

Aufrufen von POUs:

POUs können andere POUs aufrufen. Rekursionen sind allerdings nicht erlaubt.

Wenn eine einer Applikation zugehörige POU eine andere POU **nur mit deren Namen** (ohne Zusatz eines Namensraums) aufruft, gilt die folgende Reihenfolge, in der das Projekt nach der aufzurufenden POU durchsucht:

1. Aktuelle Applikation
2. Bibliotheksverwalter der aktuellen Applikation
3. POUs-Fenster
4. Biblitheksverwalter im POUs-Fenster.

Wenn es eine POU mit dem im Aufruf angegebenen Namen sowohl in einer Bibliothek des Applikations-Bibliotheksverwalters als auch als Objekt im POUs-Fenster gibt, gibt es keine Syntax nach der die POU im POUs-Fenster nur über ihren Namen aufgerufen werden kann. In diesem Fall müßte man die betreffende Bibliothek vom Applikations-Bibliotheksverwalter in den Bibliotheksverwalter im POUs-Fenster verschieben. Dann könnte das POU-Objekt im POUs-Fenster rein über seinen Namen aufgerufen werden (und wenn erforderlich, die POU der Bibliothek durch Hinzufügen des Bibliothek-Namensraums).

Hinweis: Beachten Sie die speziellen "POUs für implizite Prüfungen".

3.7.2 Programm

Ein Programm ist eine POU, die bei der Ausführung einen oder mehrere Werte liefert. Alle Werte bleiben von einer Ausführung des Programms bis zur nächsten erhalten.

Ein Programm-Objekt kann über den Befehl Objekt hinzufügen im Projekt eingefügt werden. Wenn das Programm direkt einer bestehenden Applikation zugeordnet werden soll, muss vorher das Applikationsobjekt im Geräte-Fenster markiert werden. Ansonsten wird das Programm im POUs-Fenster eingefügt und ist damit projektweit verfügbar. Wählen Sie Im 'Objekt hinzufügen'-Dialog links den Objekttyp 'POU' und dann rechts den POU-Typ 'Programm', geben Sie einen Namen für das Programm ein (<program name>) wählen Sie die gewünschte Implementationssprache. Nach Bestätigung der Einstellungen mit **Öffnen** wird das Editorfenster für das neue Programm geöffnet und Sie können mit der Implementation beginnen:

Deklaration:

Syntax:

PROGRAM <program name>

Hier folgen die Variablendeklarationen für Eingabe-, Ausgabe- und Programmvariablen, optional auch Access-Variablen.

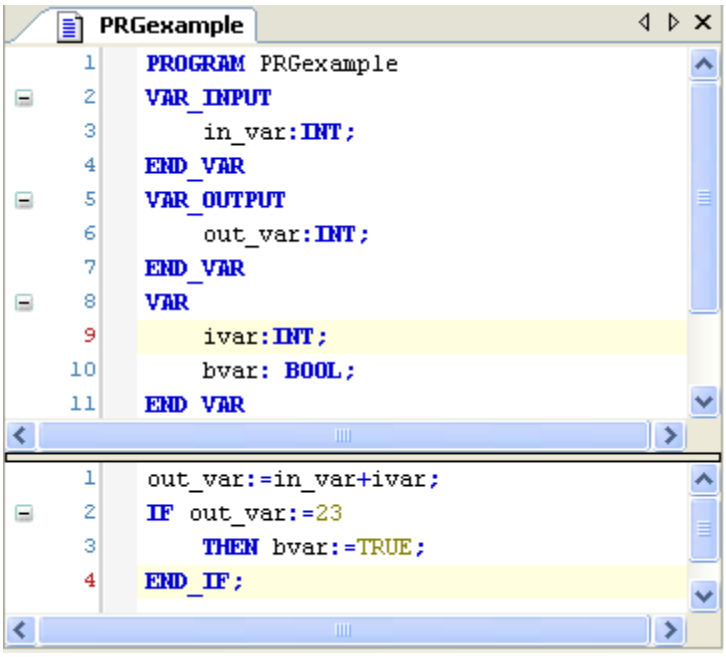


Abb. 5 Beispiel eines Programms

Programmaufrufe:

Ein Programm kann von einer anderen POU aufgerufen werden. Aber: Ein Programmaufruf in einer Funktion ist nicht erlaubt. Es gibt auch keine Instanzen von Programmen.

Wenn eine POU ein Programm aufgerufen hat und sich dadurch Werte des Programms verändert haben, bleiben diese Änderungen erhalten, bis das Programm erneut aufgerufen wird, auch wenn der erneute Aufruf durch eine andere POU erfolgt. Dies unterscheidet sich vom Aufruf eines Funktionsbausteins, wo sich nur die Werte in der jeweiligen Instanz des Funktionsbausteins ändern und die Änderungen somit nur zu beachten sind, wenn dieselbe Instanz erneut aufgerufen wird.

Wenn Sie Eingabe- und/oder Ausgabeparameter für ein Programm unmittelbar während des Programmaufrufs setzen wollen, können Sie dies innerhalb einer Klammer direkt nach dem Programmnamen tun. Für Eingabeparameter wird die Zuweisung mit ":=" angegeben, genauso wie bei der Initialisierung von Variablen im Deklarationsteil, für Ausgabeparameter wird "=>" verwendet; Sehen Sie unten ein Beispiel.

Wenn ein Programmaufruf via Eingabehilfe und Option **Mit Argumenten einfügen** im Implementationsteil eines textuellen Editors eingefügt wird, wird er automatisch gemäß der im vorangegangenen Absatz beschriebenen Syntax mit allen Parametern dargestellt. Sie können dann, müssen aber nicht, die entsprechenden Wertezuweisungen einfügen.

Beispiele für Programmaufrufe:

In AWL:

```
CAL                PRGexample          (
        in_var:=   33                  )
LD                 PRGexample.out_var
ST                 erg
```

oder mit Zuweisung der Parameter (Eingabehilfe "Mit Argumenten einfügen", siehe oben):

```
CAL                PRGexample          (
        in_var:=   33                  ,
       out_var=>   erg                 )
```

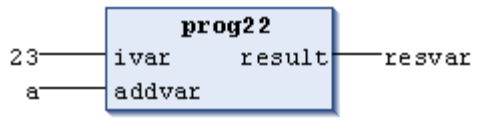
In ST (beachten Sie, dass im Unterschied zu V2.3 die **Klammern obligatorisch** sind):

```
PRGexample();
erg := PRGexample.out_var;
```

oder mit Zuweisung der Parameter (Eingabehilfe "Mit Argumenten einfügen", siehe oben):

```
PRGexample(in_var:=33, out_var=>erg );
```

In FUP:



3.7.3 Funktion

Eine Funktion ist eine POU, die bei der Ausführung genau ein Datenelement (das auch mehrelementig sein kann, wie z.B. ein Feld oder eine Struktur) liefert, und dessen Aufruf in textuellen Sprachen als ein Operator in Ausdrücken vorkommen kann.

Ein Funktionsobjekt kann über den Befehl Objekt hinzufügen im Projekt eingefügt werden. Wenn die Funktion direkt einer bestehenden Applikation zugeordnet werden soll, muss vorher das Applikationsobjekt im Geräte-Fenster markiert werden. Ansonsten wird das Programm im POUs-Fenster eingefügt. Wählen Sie Im 'Objekt hinzufügen'-Dialog links den Objekttyp 'POU' und dann rechts den POU-Typ 'Funktion', geben Sie einen Namen für die Funktion ein (<function name>), wählen Sie einen Rückgabetyp (<data type>) und die gewünschte Implementationssprache. Zum Auswählen des Rückgabetyps kann über die Schaltfläche ... die Eingabehilfe geöffnet werden. Nach Bestätigung der Einstellungen mit **Öffnen** wird das Editorfenster für die neue Funktion geöffnet und Sie können mit der Implementation beginnen:

Deklaration:

Syntax:

FUNCTION <function name> : <data type>

Hier folgen die Variablendeklarationen der Eingabe- und Funktionsvariablen.

Jeder Funktion muss ein Ergebnis zugewiesen werden, d.h. der Funktionsname wird als Ausgabeparameter verwendet.

Hinweis: Wenn eine lokale Variable in einer Funktion als RETAIN deklariert wird, hat dies keinen Effekt! Die Variable wird nicht in den Retain-Speicher geschrieben!

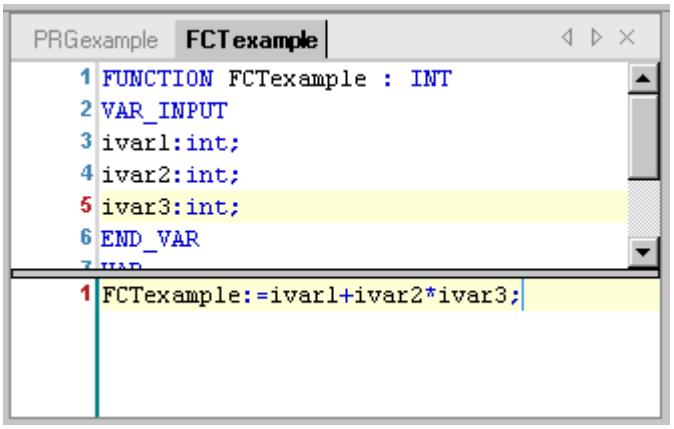


Abb. 6 Beispiel einer Funktion in ST (Diese Funktion erhält drei Eingabevariablen und gibt das Produkt der letzten zwei, addiert zu der ersten, zurück)

Funktionsaufruf:

Der Aufruf einer Funktion kann in ST als Operand in Ausdrücken verwendet werden.

In SFC kann ein Funktionsaufruf nur innerhalb von Schrittaktionen oder Transitionen erfolgen.

Funktionen enthalten (im Gegensatz zu Programmen oder Funktionsbausteinen) keine interne Statusinformation. Das heißt, Aufrufe einer Funktion mit denselben Argumenten (Eingabeparameter) liefern immer denselben Wert (Ausgabe). Deshalb dürfen Funktionen keine globalen Variablen und Adressen verwenden!.

Beispiele von Funktionsaufrufen:

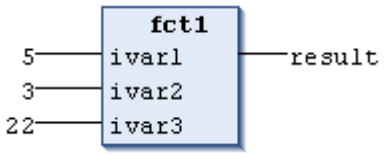
in AWL:

```
LD      5
Fct     3   ,
        22
ST      result
```

in ST:

```
result := fct1(5,3,22);
```

in FUP:



Hinweis: Im Unterschied zu PLC Designer V2.3 können in Funktionsaufrufen explizite und implizite Parameterzuweisungen nicht mehr gemischt werden. Somit ist die Reihenfolge der Parameterzuweisungen beim Aufruf nicht mehr vorgeschrieben.
Beispiel:
fun(formal1 := actual1, actual2); // -> Fehlermeldung
fun(formal2 := actual2, formal1 := actual1); // gleiche Semantik wie folgendes:
fun(formal1 := actual1, formal2 := actual2);
Allerdings muss diese unterschiedliche Behandlung der Parameterzuweisungen beachtet werden, wenn V2.3 Projekte bearbeitet werden!

Laut IEC 61131-3 Norm können Funktionen **zusätzliche Ausgänge** haben. Diese müssen im Funktionsaufruf gemäß folgender Syntax zugewiesen werden:

```
out1 => <Ausgabevariable 1> | out2 => <Ausgabevariable 2> | ...weitere Ausgabevariablen
```

Beispiel: Funktion *fun* ist mit zwei Eingabevariablen *in1* und *in2* definiert. Der Rückgabewert von *fun* wird auf die local deklarierten Ausgangsvariablen (VAR_OUTPUT) *loc1* und *loc2* geschrieben.

```
fun(in1 := 1, in2 := 2, out1 => loc1, out2 => loc2);
```

Erweiterbare Funktionen

Als Erweiterung zur Norm IEC 61131-3 können Funktionen und Methoden mit einer variablen Anzahl von Eingangsparametern desselben Typs versehen werden.

3.7.4 Funktionsblock

Ein Funktionsblock (**=Funktionsbaustein**) ist eine POU die bei der Ausführung einen oder mehrere Werte liefert. Im Gegensatz zu einer Funktion werden die Werte der Ausgabevariablen und der verwendeten internen Variablen von einer Ausführung bis zur nächsten behalten. Somit werden bei mehrmaligem Aufruf eines Funktionsbausteins mit denselben Eingabeparametern nicht unbedingt dieselben Ausgabewerte geliefert.

Zusätzlich zu der in der IEC11631-3 Norm beschriebenen Funktionalität wird auch **objektorientierte Programmierung** unterstützt und Funktionsbausteine können als Erweiterungen anderer Funktionsbausteine definiert werden, und sie können Schnittstellen-Definitionen für Methodenaufrufe enthalten. Dies bedeutet, dass beim Programmieren mit Funktionsbausteinen das Prinzip der "Vererbung" angewendet werden kann.

Ein Funktionsbaustein **wird immer über eine** Instanz aufgerufen, die eine Kopie des Funktionsbausteins ist.

Ein Funktionsbaustein kann über den Befehl Objekt hinzufügen im Projekt eingefügt werden. Wenn er direkt einer bestehenden Applikation zugeordnet werden soll, muss vorher das Applikations-Objekt im Geräte-Fenster markiert werden. Ansonsten wird der Baustein im POUs-Fenster eingefügt.

Wählen Sie Im 'Objekt hinzufügen'-Dialog links den Objekttyp 'POU' und dann rechts den POU-Typ 'Funktionsblock', geben Sie einen Namen für den Baustein ein (<Funktionsblockname>) und wählen Sie die gewünschte Implementationssprache.

Zusätzlich können die folgenden Optionen aktiviert werden:

Erweitert: Geben Sie den Namen eines anderen Funktionsbausteins aus dem Projekt ein, der als Basis für den vorliegenden verwendet werden soll.

Implementiert: Geben Sie den oder die Namen von im Projekt definierten Schnittstellen an, die im vorliegenden Funktionsbaustein implementiert werden sollen. Mehrere Schnittstellennamen können durch Kommata getrennt angegeben werden. Im Feld **Methodenimplementierungssprache** kann für alle durch die Schnittstellenimplementation angelegten Methodenobjekte - unabhängig von der für den Funktionsblock eingestellten - die gewünschte Programmiersprache ausgewählt werden.

Nach Bestätigung der Einstellungen mit **Öffnen** wird das Editorfenster für den neuen Funktionsbaustein geöffnet und Sie können mit der Implementation beginnen:

Deklaration:

Syntax:

FUNCTION_BLOCK <Funktionsblockname> | EXTENDS <Funktionsblockname> | IMPLEMENTS <kommaseparierte Liste von Schnittstellennamen>

Hier folgen die Variablendeklarationen.

Hinweis: Im Unterschied zu PLC Designer V2.3 ist "FUNCTIONBLOCK" kein gültiges Schlüsselwort mehr!

Beispiel:

FBexample in der unten gezeigten Abbildung hat zwei Eingabevariablen inp1 und inp2 und zwei Ausgabevariablen out1 und out2. Out1 ist die Summe der beiden Eingänge, out2 ist das Ergebnis einer Prüfung auf Gleichheit.

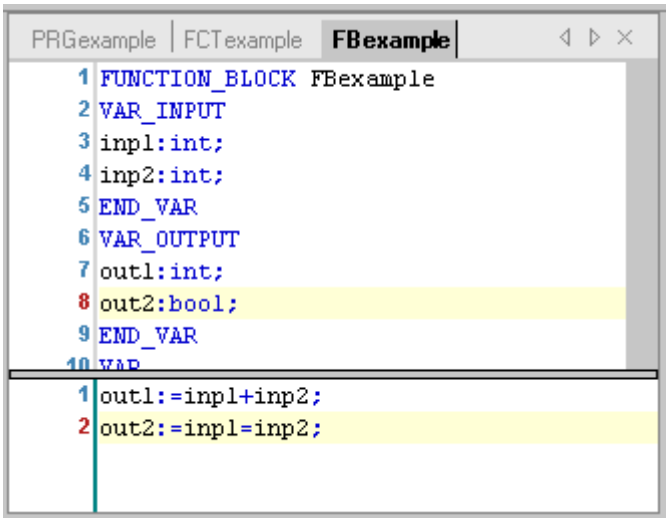


Abb. 7 Beispiel eines Funktionsbausteins in ST

Instanz eines Funktionsblocks

Funktionsblöcke (Funktionsbausteine) werden immer über eine Instanz aufgerufen, die eine Kopie des Funktionsblocks ist.

Jede Instanz hat einen Bezeichner (Instanzname) und eine Datenstruktur, die ihre Eingabe-, Ausgabe- und internen Variablen beinhaltet.

Instanzen werden wie Variablen lokal oder global deklariert, wobei der Name des Funktionsblocks als Typ des Bezeichners angegeben wird.

Syntax für die Deklaration einer Funktionsblock-Instanz:

<Bezeichner>:<Funktionsblockname>;

Beispiel

Deklaration (z.B. im Deklarationsteil eines Programms) einer Instanz "INSTANCE" des Funktionsblocks "FUB":

```
INSTANCE: FUB;
```

Instanzdeklarationen können in den Deklarationsteilen von Funktionsbausteinen und Programmen vorgenommen werden. In Funktionen sind sie nicht zulässig.

Aufruf eines Funktionsblocks

Funktionsblöcke werden immer über eine Funktionsblock-Instanz aufgerufen. Die Instanz muss lokal oder global deklariert sein (<Instanzname>).

Auf die gewünschte Funktionsblock-Variable (<Variablenname>) kann dann gemäß der folgenden Syntax zugegriffen werden:

Syntax:

<Instanzname>.<Variablenname>

Beachten Sie folgendes:

- Nur auf die Eingabe- und Ausgabevariablen eines Funktionsbausteins kann von außerhalb der Funktionsblock-Instanz zugegriffen werden, nicht auf die internen Variablen.
- Der Zugriff auf eine Funktionsblock-Instanz ist auf die POU begrenzt, in der sie deklariert ist, außer sie ist global deklariert.
- Beim Aufruf der Instanz können den Funktionsblock-Parametern die gewünschten Werte zugewiesen werden. Siehe unten.
- Bitte beachten Sie, dass die InOutVariablen (VAR_IN_OUT) eines Funktionsblocks als Pointer übergeben werden. Siehe unten.
- In SFC können Funktionsblock-Aufrufe nur in Schritten erfolgen.
- Der Name einer Funktionsblock-Instanz kann als Eingabeparameter für eine Funktion oder einen anderen Funktionsblock verwendet werden.
- Alle Werte eines Funktionsblocks bleiben bis zur nächsten Ausführung des Bausteins erhalten. Deshalb liefern Funktionsblock-Aufrufe nicht immer dieselben Ausgabewerte, auch wenn dieselben Eingabewerte verwendet werden!

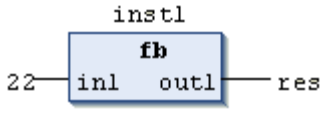
Hinweis: Wenn mindestens einer der Funktionsblock-Variablen eine remanente Variable ist, wird die gesamte Instanz im Retain-Bereich gespeichert.

Beispiele für Zugriffe auf Funktionsblock-Variablen

Annahme: Funktionsblock *fb* hat eine Eingabevariable *in1* vom Typ INT. Sehen Sie im folgenden die Aufrufe dieser Variable aus dem Programm prog heraus. Deklaration und Implementation in ST:

```
PROGRAM prog
VAR
inst1:fb;
END_VAR

inst1.in1:=22;  (* fb wird aufgerufen und der Wert 22 wird der Eingabevariablen in1 zugewiesen *)
inst1();        (* fb wird aufgerufen, dies ist nötig für den folgenden Zugriff auf die Ausgabevariable *)
res:=inst1.outl; (* die Ausgabevariable out1 von fb wird gelesen *)
```

Beispiel in FBD:



Zuweisung von Parametern beim Aufruf:

In den textuellen Sprachen **AWL** und **ST** können Eingabe- und/oder Ausgabeparameter direkt beim Aufruf des Funktionsblocks gesetzt werden. Die Werte können den Parametern innerhalb einer Klammer direkt nach dem Funktionsblock-Namen zugewiesen werden. Für Eingabeparameter wird die Zuweisung mit ":=" angegeben, genauso wie bei der Initialisierung von Variablen im Deklarationsteil, für Ausgabeparameter wird "=>" verwendet; Sehen Sie im folgenden ein Beispiel.

Beispiel, Aufruf mit Zuweisungen:

In diesem Beispiel wird ein Timer-Funktionsblock (Instanz "CMD_TMR") mit Zuweisungen für die Parameter IN und PT aufgerufen. Danach wird die Ausgabevariable Q des Timers der Variable A zugewiesen. Die Ausgabevariable wird gemäß der Syntax <FBInstanz>.<FB-Ausgabevariable> angesprochen:

```
CMD_TMR(IN := %IX5, PT := 300);
A:=CMD_TMR.Q
```

Wenn die Instanz via Eingabehilfe und Option **Mit Argumenten einfügen** im Implementationsteil eines textuellen Editors eingefügt wird, sie automatisch gemäß der oben beschriebenen Syntax mit allen Parametern dargestellt. Sie können dann, müssen aber nicht, die entsprechenden Wertezuweisungen einfügen.

Für das oben beschriebene Beispiel würde der Aufruf dann folgendermaßen dargestellt:

Beispiel, Einfügen über Eingabehilfe, mit Argumenten:

```
CMD_TMR(in:=, pt:=, q=>)
-> Zuweisungen entsprechend ergänzen, z.B. :
CMD_TMR(in:=bvar, pt:=t#200ms, q=>bres);
```

Erweiterung eines Funktionsblocks

Objektorientierte Programmierung wird durch die Möglichkeit unterstützt, einen Funktionsblock von einem anderen Funktionsblock abzuleiten. Das heißt, ein Funktionsblock kann einen anderen erweitern und erhält dabei die Eigenschaften des Basisbausteins zusätzlich zu seinen eigenen.

Die Erweiterung erfolgt in der Deklaration mit Hilfe des Schlüsselwortes EXTENDS. Die "Erweitert" Option kann bereits beim Hinzufügen eines Funktionsbausteins im Projekt über den Objekt hinzufügen-Dialog aktiviert werden:

Syntax:

FUNCTION_BLOCK <Funktionsblockname> EXTENDS <Funktionsblockname>

Hier folgen die Variablendeklarationen.

Beispiel:

Definition von Funktionsblock fbA:

```
FUNCTION_BLOCK fbA
VAR_INPUT
x:int;
....
```

Definition von Funktionsblock fbB:

```
FUNCTION_BLOCK fbB EXTENDS fbA
VAR_INPUT
ivar:int;
....
```

Die Erweiterung mit Hilfe von EXTENDS bedeutet:

- fbB enthält alle Daten und Methoden, die von fbA definiert sind. Eine Instanz von fbB kann nun in jedem Kontext verwendet werden, in dem ein Funktionsblock vom Typ fbA erwartet wird.
- fbB darf die Methoden, die in fbA definiert sind, überschreiben. Das bedeutet: fbB kann eine Methode mit demselben Namen, denselben Eingängen und demselben Ausgang definieren, wie sie von fbA definiert ist.
- fbB darf keine Funktionsblock-Variablen mit denselben Namen, wie in fbA verwendet, enthalten. In diesem Fall meldet der Compiler einen Fehler. Einzige Ausnahme: Ist eine Variable in fbA als VAR_TEMP deklariert, so darf fbB eine Variable desselben Namens definieren, kann auf die Variable des Basis-Funktionsblock jedoch nicht mehr zugreifen.
- fbA Variablen und Methoden können innerhalb des Gültigkeitsbereichs von fbB durch Verwendung des Schlüsselwortes "**SUPER**" (SUPER^.<method>).direkt angesprochen werden.

Hinweis: Mehrfache Vererbung ist nicht erlaubt!

Schnittstellen implementieren

Im Sinne der **objektorientierten Programmierung** kann ein Funktionsblock Schnittstellen implementieren, die die Verwendung von Methoden ermöglichen:

Syntax:

FUNCTION_BLOCK <Funktionsblockname> IMPLEMENTS <Name Schnittstelle 1>|,<Name Schnittstelle 2>, ..., <Name Schnittstelle n>

Ein Funktionsblock, der eine Schnittstelle implementiert, muss alle Methoden, die in dieser Schnittstelle definiert sind, enthalten. Das heißt, der Name, die Eingänge und der Ausgang der Methoden muss jeweils genau identisch sein.

Dazu werden beim Anlegen eines neuen Funktionsblocks, der eine Schnittstelle implementiert, automatisch alle in der Schnittstelle definierten Methoden und Eigenschaften auch unterhalb des neuen Funktionsblock eingehängt. Momentan werden allerdings nachträglich an der Schnittstelle vorgenommene Änderungen, z.B. das Hinzufügen weiterer Methoden, noch nicht automatisch in den betroffenen Funktionsblöcken nachgezogen. Dies muss noch explizit mit dem Befehl 'Schnittstellen implementieren...' durchgeführt werden.

Beispiel:

INTERFACE I1 enthält Methode GetName:

```
METHOD GetName : STRING
```

Funktionsblock A und B implementieren jeweils Schnittstelle I1:

```
FUNCTION_BLOCK A IMPLEMENTS I1
FUNCTION_BLOCK B IMPLEMENTS I1
```

Somit muss in beiden Funktionsblöcke die Methode "GetName" vorhanden sein und wird auch automatisch beim Anlegen der FBs unterhalb von diesen im Objektbaum eingehängt.

Betrachten Sie nun die Deklaration einer Variable vom Typ der Schnittstelle I1:

```
FUNCTION DeliverName : STRING
VAR_INPUT
 I_i : I1;
END_VAR
```

Diese Eingabevariable kann alle Funktionsblöcke empfangen, die Schnittstelle I1 implementieren.

Beispiele für Funktionsaufrufe:

```
DeliverName(I_i := A_instance); // Aufruf mit Instanz von Funktionsblock A
DeliverName(I_i := B_instance); // Aufruf mit Instanz von Funktionsblock B
```

Beachten Sie, dass eine Variable vom Typ einer Schnittstelle...

... die Schnittstelle eines Funktionsblocks zugewiesen bekommen muss, bevor eine Methode über sie aufgerufen werden kann.

... immer eine Referenz der zugewiesenen Funktionsblock-Instanz ist. Somit bewirkt ein Aufruf der Schnittstellen-Methode einen Aufruf der Funktionsblock-Implementierung. Sobald eine Referenz erstellt ist, wird im Online-Modus die zugehörige Adresse angezeigt. Andernfalls, falls noch keine Referenz erzeugt wurde, wird hier der Wert 0 angezeigt.

Beispiel:

Im Implementierungsteil der oben beschriebenen Funktion "DeliverName":

```
DeliverName := I_i.GetName();  // in diesem Fall hängt es vom "tatsächlichen" Typ von I_i ab, ob A.GetName oder B.GetName aufgerufen wird.
```

Hinweis: Beachten Sie auch die Möglichkeit, einen Funktionsblock mit Hilfe von Schlüsselwort EXTENDS in der Deklaration zu erweitern.

Methodenaufruf

Objektorientierte Programmierung mit Funktionsbausteinen (Funktionsblöcken) wird - neben der Möglichkeit der Erweiterung via EXTENDS - auch durch die mögliche Verwendung von Schnittstellen und Vererbung unterstützt. Dies erfordert dynamisch aufgelöste Methodenaufrufe, auch "virtuelle Funktionsaufrufe" genannt.

Virtuelle Funktionsaufrufe benötigen etwas mehr Zeit als normale Funktionsaufrufe und werden verwendet, wenn::

- ein Aufruf über einen Pointer auf einen Funktionsblock erfolgt (z.B. "pfub^.method")
- eine Methode einer Schnittstellen-Variable aufgerufen wird (z.B. "interface1.method")
- eine Methode eine andere Methode desselben Funktionsblocks aufruft

Virtuelle Funktionsaufrufe erlauben, dass ein-und derselbe Aufruf in einem Programm-Quellcode während der Laufzeit verschiedene Methoden aufruft.

Gemäß der Norm IEC 61131-3 können Methoden wie "normale" Funktionen **zusätzliche Ausgänge** haben. Diese müssen beim Methoden-Aufruf gemäß folgender Syntax zugewiesen werden:

<method>(in1:=<value> |, further input assignments, out1 => <output variable 1> | out2 => <output variable 2> | ...further output variables)

Dies bewirkt, dass der Ausgang der Methode wie im Aufruf definiert auf die lokal deklarierten Ausgabevariablen geschrieben wird.

Beispiel:

Annahme: Funktionsblöcke *fub1* und *fub2* EXTEND (erweitern) Funktionsblock *fubbase* und IMPLEMENT (implementieren) Schnittstelle *interface1*. Die Methode *method1* ist enthalten. Mögliche Verwendung der Interfaces und Aufruf der Methode:

```
VAR_INPUT
  b : BOOL;
END_VAR
VAR
  pInst : POINTER TO fubbase;
  instBase : fubbase;
  inst1 : fub1;
  inst2 : fub2;
  interfaceRef : interface1;
END_VAR

IF b THEN
  interfaceRef := inst1;   (* Interface1 für fub1 wird verwendet *)
  pInst := ADR(instBase);
ELSE
  interfaceRef := inst2;   (* Interface1 für fub2 wird verwendet *)
  pInst := ADR(inst1);
END_IF
pInst^.method1();      (* Wenn b=true, wird fubbase.method1 aufgerufen,
ansonsten fub1.method1 *)
interfaceRef.method1(); (* Wenn b=true, wird fub1.method1 aufgerufen,
ansonsten fub2.method1 *)
```

Nehmen wir nun an, dass *fubbase* zwei Methoden *method1* und *method2* enthält und *fub1 method2*, aber nicht *method1* überschreibt:

method1 wird wie im obigen Beispiel folgendermaßen aufgerufen:

```
pInst^.method1(); (* Wenn b=true, wird fubbase.method1 aufgerufen, ansonsten fub1.method1 *)
```

Aufruf mit THIS-Pointern:

Betrachten wir den Implementationsteil von *method1*:

```
METHOD method1 : BOOL
VAR_INPUT
END_VAR
method1 := method2();
```

Für den Aufruf von *fubbase.method1* mit THIS-Pointern gilt dann folgendes:

Wenn THIS vom Typ *fubbase* ist, wird *fubbase.method2* aufgerufen. Wenn THIS vom Typ *fub1* ist, wird *fub1.method2* aufgerufen.

3.7.5 DUT / Datentyp

Zusätzlich zu den Standard-Datentypen kann der Anwender eigene Datentypen definieren. Strukturen, Enumerationen und Referenzen können als Datentyp-Objekte (**DUT**s=Data Unit Types) im DUT Editor angelegt werden.

Ein Datentyp-Objekt kann über den Befehl Objekt hinzufügen im Projekt eingefügt werden. Wenn es direkt einer bestehenden Applikation zugeordnet werden soll, muss vorher das Applikationsobjekt im Geräte-Fenster markiert werden. Ansonsten wird die DUT im POUs-Fenster eingefügt. Wählen Sie Im 'Objekt hinzufügen'-Dialog links den Objekttyp 'DUT' und geben Sie einen Namen für den neuen Datentyp ein (<DUT name>). Nach Bestätigung der Einstellungen mit **Öffnen** wird das Datentyp-Editorfenster geöffnet und Sie können mit der Implementation beginnen.

Für die **objektorientierte Programmierung** können sie bei Datentypen das Prinzip der Vererbung anwenden. Im 'Objekt hinzufügen'-Dialog kann festgelegt werden, dass der Datentyp einen **anderen Datentyp erweitern** soll, der bereits im Projekt definiert ist. Das bedeutet, daß die Definitionen des erweiterten DUTs automatisch im vorliegenden gültig sind. Die Erweiterung wird über Option **Erweitert:** aktiviert und der Name der zu erweiternden DUT angegeben.

Nach Bestätigung der Einstellungen mit **Öffnen** wird der DUT-Editor für den neuen Datentyp geöffnet und Sie können mit der Implementation beginnen:

Deklaration:

Syntax:

TYPE <Bezeichner> : <DUT Komponenten-Deklaration>

END_TYPE

Die Komponenten-Deklaration hängt vom gewählten Typ ab, z.B. eine Struktur oder eine Enumeration.

Beispiel:

Sehen Sie im folgenden zwei DUTs, die die Strukturen struct1 und struct2 definieren; struct2 erweitert struct1, was bedeutet, daß "struct2.a" verwendet werden kann, um auf Variable a zuzugreifen.

```
TYPE struct1 :
STRUCT
 a:INT;
 b:BOOL;
END_STRUCT
END_TYPE

TYPE struct2 EXTENDS struct1 :
STRUCT
 c:DWORD;
 d:STRING;
END_STRUCT
END_TYPE
```

3.7.6 Methode

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Objektorientierte Programmierung wird durch die mögliche Verwendung von Methoden unterstützt, die eine Abfolge von Anweisungen enthalten. Wie eine Funktion ist eine Methode nicht eine unabhängige POU, sondern **muss einem** Funktionsblock **zugewiesen sein**. Sie kann als Funktion betrachtet werden, die eine Instanz des jeweiligen Funktionsbausteins beinhaltet.

Ebenso der **objektorientierten Programmierung** dienlich ist die Verwendung von Schnittstellen, zur projektinternen Organisation von Methoden. Eine Schnittstelle in diesem Zusammenhang ist eine Sammlung von Methoden-"Prototypen". D.h. eine Methode, die einer Schnittstelle zugewiesen ist, enthält nur einen Deklarationsteil, aber keine Implementation. Die Implementation wird im Funktionsblock vorgenommen, der die Schnittstelle implementiert und die Methode verwendet.

Der Vorteil: Derselbe Methodenaufruf kann in allen Funktionsblöcken, die dieselbe Schnittstelle implementieren, verwendet werden. D.h. der Aufruf kann für verschiedene Zwecke verwendet werden. Eine Methode aufzurufen bedeutet also nur grob anzugeben, welcher Zweck erreicht werden soll, es hängt dann vom jeweils betroffenen Funktionsblock ab, welche Anweisungen (Implementation) konkret und im Detail ausgeführt werden um dem Zweck zu genügen.

Hinweis: Beim Kopieren oder Verschieben einer Methode von einer POU zu einer Schnittstelle werden die enthaltenen Implementationen automatisch gelöscht. Beim Kopieren oder Verschieben von einer Schnittstelle zu einer POU wird der Anwender aufgefordert, die gewünschte Implementierungssprache anzugeben.

Eine Methode kann über den Befehl Objekt hinzufügen im Projekt eingefügt werden, wenn der Funktionsbaustein, dem sie zugeordnet werden soll, im POUs- oder Geräte-Fenster ausgewählt ist. Wählen Sie im 'Objekt hinzufügen'-Dialog links den Objekttyp 'Methode' und geben Sie wie bei einer Funktion einen Namen (<Methodenname>) und einen Rückgabetyp (<data type>) ein. Zum Auswählen des Rückgabetyps kann über die Schaltfläche [...] die Eingabehilfe geöffnet werden. Nach Bestätigung der Einstellungen mit Öffnen wird das Editorfenster für die Methode geöffnet.

Deklaration:

Syntax:

METHOD <method name> : <return data type>

VAR_INPUT
END_VAR

Aufruf:

Methodenaufrufe werden auch "virtuelle Funktionsaufrufe" genannt.

Hinweis:
Alle Daten einer Methode sind temporäre Daten und nur während der Ausführung einer Methode gültig (Stack-Variablen).
Im Implementationsteil einer Methode ist der Zugriff auf die Funktionsblock-Instanz-Variablen erlaubt.
THIS pointer: Der Bezeichner "THIS" kann verwendet werden, um direkt auf die implizite Instanz des betreffenden Funktionsblocks zu zeigen, die automatisch verfügbar ist. Beachten Sie, dass eine lokal deklarierte Variable eventuell eine Funktionsblock-Variable verschattet. Im Falle des oben bei "Deklaration" gezeigten Beispiels etwa würde "THIS^.x" sich nicht auf den Methoden-Eingang x, sondern auf die Funktionsblock-Variable x beziehen.
Auf VAR_IN_OUT oder VAR_TEMP-Variablen des Funktionsblocks kann nicht in einer Methode zugegriffen werden!
Methoden, die in einer Schnittstelle definiert sind, dürfen nur Eingabe-, Ausgabe- und EinAusgabe-Variablen definieren, aber keine Implementation enthalten.
Methoden können wie Funktionen zusätzliche Ausgänge erhalten. Diese müssen im Methodenaufruf zugewiesen werden.

Spezielle Methoden eines Funktionsblocks:

Init Methode: Eine Methode mit Namen "FB_init" wird immer implizit deklariert, kann aber auch explizit deklariert werden. Sie enthält Initialisierungs-Code für den Funktionsblock wie er im Deklarationsteil des Funktionsblocks definiert ist. Siehe weiter unter FB_init Methode.

Reinit Methode: Wenn eine Methode mit Namen "FB_reinit" deklariert ist, wird diese aufgerufen, wenn eine Instanz des Funktionsblocks kopiert wird. Sie startet eine Re-Initialisierung des neuen Instanz-Moduls. Siehe weiter unter FB_init_Methode.

Exit Methode: Wenn eine Methode mit Namen "FB_exit" deklariert ist, wird diese für jede Instanz des Funktionsblocks vor einem Download, einem Reset oder bei einem Online Change aufgerufen. Siehe weiter unter FB_exit Methode.

Properties (Eigenschaften) : Siehe weiter unter Eigenschaft.

Aufruf einer Methode, auch wenn die Applikation im STOP-Status ist

In der Gerätebeschreibung kann definiert sein, dass eine bestimmte Methode von einer bestimmten Funktionsblockinstanz (eines Bibliotheksbausteins) immer taskzyklisch aufgerufen werden soll. Wenn diese Methode die folgenden Eingabeparameter enthält, wird sie auch abgearbeitet, wenn die aktive Applikation gerade im STOP-Status ist:

```
VAR_INPUT
pTaskInfo : POINTER TO DWORD;
pApplicationInfo: POINTER TO _IMPLICIT_APPLICATION_INFO;
END_VAR
```

Nun kann der Applikationsstatus über pApplicationInfo abgefragt und entsprechende Anweisungen können programmiert werden (z. B.: IF pApplicationInfo^.state=RUNNING THEN <Anweisungen> END_IF).

3.7.7 Eigenschaft

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Ein Baustein vom Typ "**Eigenschaft**" kann über den Befehl Objekt hinzufügen einem Programm oder Funktionsblock zugewiesen, d.h. im Objektbaum unterhalb diesem eingefügt werden. Dabei sind der **Rückgabetyp** und die gewünschte **Implementationssprache** anzugeben.

Eine solche "Eigenschaft" enthält zwei spezielle Methoden: Sie werden im Objektbaum automatisch unterhalb des Eigenschaften-Objekts eingehängt:

- die "Set" Methode wird aufgerufen,wenn auf die Eigenschaft schreibend zugegriffen wird, d.h. der Name der Eigenschaft als Eingabeparameter verwendet wird.
- die "Get" Methode wird aufgerufen, wenn auf die Eigenschaft lesend zugegriffen wird, d.h. der Name der Eigenschaft als Ausgabeparameter verwendet wird.

Beispiel:

Funktionsbaustein FB1 verwendet eine lokale Variable "milli". Diese Variable wird durch die Eigenschaften Get und Set bestimmt:

Get:

```
seconds := milli / 1000;
```

Set:

```
milli := seconds * 1000;
```

Auf die Eigenschaft kann geschrieben werden (Set Methode), z.B. mit "fbinst.seconds := 22;" (fbinst ist die Instanz von FB1).

Die Eigenschaft kann gelesen werden (Get Methode), z. B. mit "testvar := fbinst.seconds;".

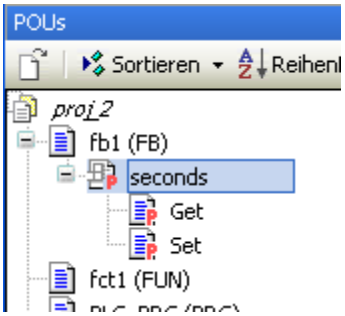


Abb. 8 Beispiel: Eigenschaft "seconds", dem Funktionsbaustein FB1 zugewiesen

Eine Eigenschaft kann zusätzliche lokale Variablen enthalten, aber keine zusätzlichen Eingänge und - im Gegensatz zu einer Funktion oder Methode - keine zusätzlichen Ausgänge.

Hinweis: Beim Kopieren oder Verschieben von Eigenschaften von einer POU zu einer Schnittstelle werden die enthaltenen Implementationen automatisch gelöscht. Beim Kopieren oder Verschieben von einer Schnittstelle zu einer POU wird der Anwender aufgefordert, die gewünschte Implementierungssprache anzugeben.

Monitoring einer Eigenschaft

Eine Eigenschaft kann im Online-Modus entweder mit Hilfe des Inline Monitoring oder mit Hilfe einer Watch-Liste überwacht werden. Notwendige Voraussetzung für das Monitoren einer Eigenschaft ist, dass der Eigenschaft das Attribut '{attribute 'monitoring':='variable'}' vorangestellt wird.

3.7.8 Schnittstelle

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Die Verwendung von Schnittstellen ist ein Mittel der **objektorientierten Programmierung**.

Schnittstellen können verwendet werden um Methoden zu verwalten, die von Funktionsblöcken implementiert werden. Eine "Schnittstelle" ist eine POU, die eine Sammlung von Methoden-Prototypen beschreibt. "Prototyp" heißt, dass nur Deklarationen, aber keine Implementation enthalten ist. Ein Funktionsblock kann eine Schnittstelle implementieren, was bedeutet, dass grundsätzlich alle in der Schnittstelle angegebenen Methoden im Funktionsblock verfügbar sein müssen.

Der Vorteil: Die Methodenaufrufe sind fest definiert und somit einheitlich für alle Funktionsblöcke, die dieselbe Schnittstelle implementieren. Innerhalb eines Funktionsblocks müssen die Methoden dann mit Implementations-Code gefüllt werden.

Folgendes ist zu beachten:

- Es ist nicht erlaubt, Variablen innerhalb einer Schnittstelle zu deklarieren. Eine Schnittstelle hat keinen Implementationsteil und keine Aktionen. Nur eine Sammlung von Methoden ist definiert und diese dürfen nur Eingabe-, Ausgabe- und EinAusgabevariablen definieren.
- Variablen, die mit dem Typ einer Schnittstelle deklariert sind, werden immer als Referenzen behandelt.
- Einem Funktionsblock, der eine Schnittstelle implementiert, müssen Methoden zugewiesen sein, die genau so benannt sind wie in der Schnittstelle und die genau dieselben Eingabe-, Ausgabe- und EinAusgabevariablen enthalten.

Hinweis: Beim Kopieren oder Verschieben von Methoden oder Eigenschaften von einer POU zu einer Schnittstelle werden die enthaltenen Implementationen automatisch gelöscht. Beim Kopieren oder Verschieben von einer Schnittstelle zu einer POU wird der Anwender aufgefordert, die gewünschte Implementierungssprache anzugeben.

Einfügen:

Eine Schnittstelle kann über den Befehl Objekt hinzufügen im Projekt eingefügt werden. Wenn sie direkt einer bestehenden Applikation zugeordnet werden soll, muss vorher das Applikationsobjekt im Geräte-Fenster markiert werden. Ansonsten wird der Baustein im POUs-Fenster eingefügt. Wählen Sie Im 'Objekt hinzufügen'-Dialog links den Objekttyp 'Schnittstelle' und geben Sie einen Namen für die Schnittstelle ein (<Schnittstellenname>).

Optional kann die Option **Erweitert:** aktiviert werden, wenn die Schnittstelle eine andere erweitern soll, was bedeutet, dass deren Methoden-Definitionen automatisch zusätzlich zu den lokal definierten gültig sind.

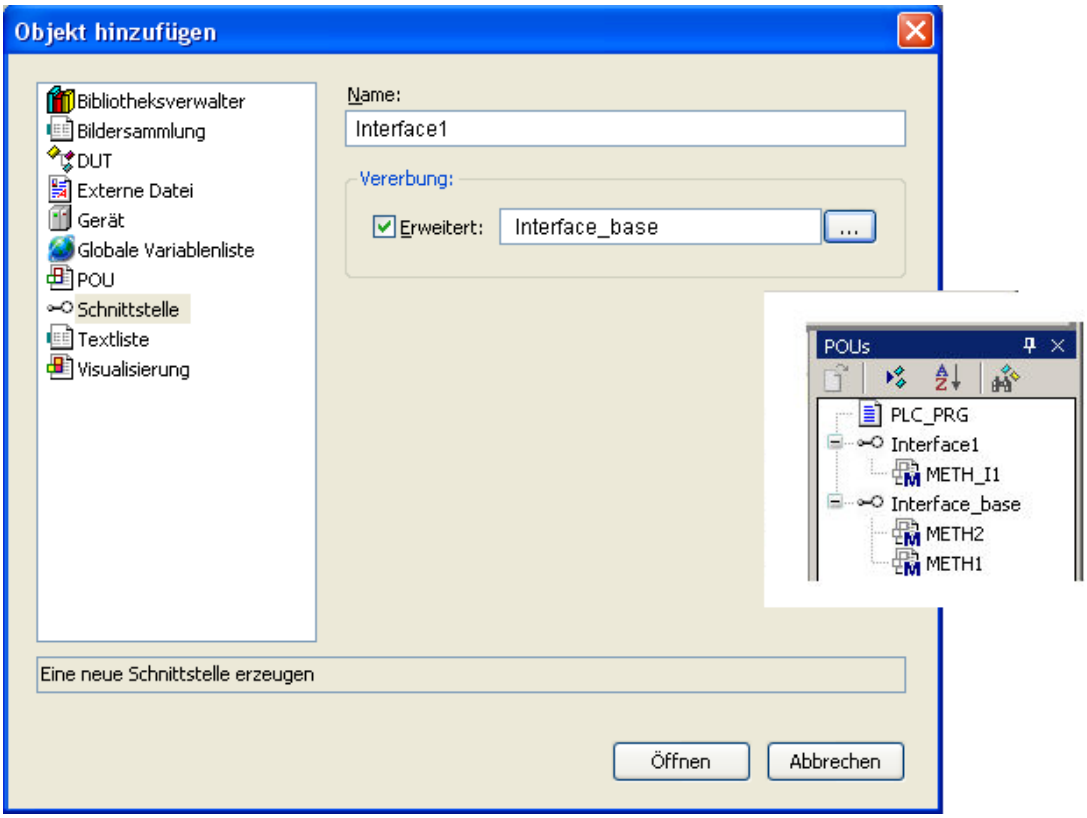


Abb. 9 Beispiel für das Erweitern einer Schnittstelle

Wenn Schnittstelle "Interface1" die Schnittstelle "Interface_base" erweitert, sind alle von Interface_base beschriebenen Methoden auch in Interface1 verfügbar.

Nach Bestätigung der Einstellungen mit **Öffnen** wird das Editorfenster für die neue Schnittstelle geöffnet.

Deklaration:

Syntax:

INTERFACE <Schnittstellenname>

bzw. für eine Schnittstelle, die eine andere erweitert:

INTERFACE <Name der neuen Schnittstelle> EXTENDS <Name der Basis-Schnittstelle>

Beispiel:

INTERFACE interface1 EXTENDS interface_base

Hinzufügen der gewünschten Sammlung von Methoden

Um die Definition der Schnittstelle zu vervollständigen, müssen die gewünschten Methoden hinzugefügt werden. Dazu wird das Schnittstellen-Objekt im POUs oder Geräte-Fenster ausgewählt und über den Befehl 'Objekt hinzufügen'-Dialog der Schnittstelle eine Methode zugewiesen. Fügen Sie alle gewünschten Methoden hinzu und beachten Sie dabei, dass die Methoden an dieser Stelle nur die Deklarationen von Eingabe-, Ausgabe- und EinAusgabevariablen, aber keine Implementation enthalten dürfen.

3.7.9 Aktion

Aktionen können über den 'Objekt hinzufügen'-Dialog definiert und Funktionsblöcken und Programmen hinzugefügt werden. Eine Aktion enthält weiteren Implementations-Code, der in einer anderen Sprache geschrieben sein kann als die "Basis"-Implementation.

Jede Aktion erhält einen Namen.

Eine Aktion arbeitet mit den Daten des Funktionsbausteins oder des Programms, dem sie zugewiesen ist. Es verwendet dessen Eingabe/Ausgabe-und lokale Variablen und enthält **keine eigenen Deklarationen**.

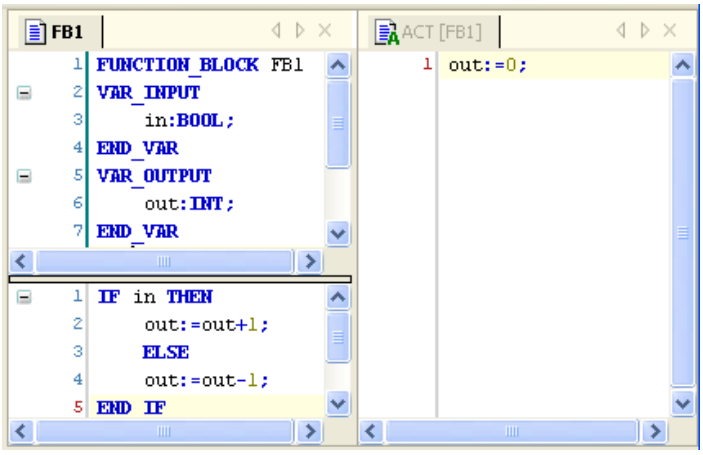


Abb. 10 Beispiel einer Aktion eines Funktionsbausteins

In diesem Beispiel erhöht jeder Aufruf des Funktionsblocks "FB1" die Ausgabevariable "out", abhängig vom Wert der Eingabevariable "in". Ein Aufruf der Aktion 'Reset' des Funktionsblocks setzt die Ausgabevariable "out" zurück auf Null. In beiden Fällen wird auf dieselbe Variable "out" geschrieben.

Eine Aktion kann über den Befehl Objekt hinzufügen im Projekt eingefügt werden, wenn der Funktionsbaustein oder das Programm, dem sie zugeordnet werden soll, im POUs- oder Geräte-Fenster ausgewählt ist. Wählen Sie Im 'Objekt hinzufügen'-Dialog links den Objekttyp 'Aktion' und geben einen Namen und die gewünschte Implementationssprache ein.

Aktionsaufruf:

Syntax:

<Programmname>.<Aktionsname>

or

<FBinstanzname>.<Aktionsname>

Beachten Sie die Notation in FUP (siehe Beispiel unten)!

Wenn eine Aktion innerhalb des Funktionsblocks oder Programms aufgerufen werden soll, dem sie zugewiesen ist, genügt es nur den Aktionsnamen anzugeben.

Beispiele für Aufrufe der im obigen Beispiel gezeigten Aktion von einer anderen POU aus:

Deklaration für alle Beispiele:

```
PROGRAM PLC_PRG
VAR
 Inst : Counter;
END_VAR
```

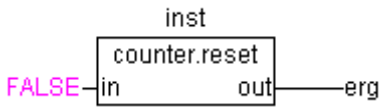
Aufruf von Aktion 'Reset' in einem anderen Baustein, der in **AWL** programmiert ist:

```
CAL Inst.Reset(In := FALSE)
LD Inst.out
ST  ERG
```

Aufruf von Aktion 'Reset' in einem anderen Baustein, der in **ST** programmiert ist:

```
Inst.Reset(In := FALSE);
Erg := Inst.out;
```

Aufruf von Aktion 'Reset' in einem anderen Baustein, der in **FUP** programmiert ist:



Hinweis: Eine Aktion ist ein wesentliches Element der Ablaufsprache. Die IEC Norm kennt nur die Aktionen der Ablaufsprache.

3.7.10 Externe Funktionen, Funktionsblöcke, Methoden

Für externe Funktionen, Funktionsblöcke oder Methoden wird vom Programmiersystem kein Code generiert.

Um einen externen Baustein anzulegen müssen die folgenden Schritte durchgeführt werden:

1. Fügen Sie die gewünschte POU im POUs Fenster ins Projekt ein wie ein anderes, internes Objekt und definieren Sie die entsprechenden Eingabe- und Ausgabevariablen.

Hinweis: Lokale Variablen müssen in "externen" Funktionsblöcken deklariert sein , aber in externen Funktionen oder Methoden nicht deklariert sein dürfen! Ebenso beachten Sie bitte, daß VAR_STAT-Variablen im Laufzeitsystem nicht verwendet werden können!

2. Definieren Sie die POU als "externe" POU: Wählen Sie dazu das POU Objekt im POUs Fenster aus und öffnen Sie den Eigenschaften-Dialog über Befehl 'Eigenschaften' im Kontextmenü. Öffnen Sie dort Registerblatt 'Übersetzen' und aktivieren Sie Option 'Externe Implementierung (Spätes Binden im Laufzeitsystem)'

Im Laufzeitsystem muss eine äquivalente Funktion, ein äquivalenter Funktionsblock bzw. ein äquivalentes Programm implementiert sein. Beim Programm-Download wird dann für jede "externe" POU der äquivalente Baustein im Laufzeitsystem gesucht und - wenn er gefunden wird - ge"linked".

3.7.11 Globale Variablenliste - GVL

Eine Globale VariablenListe ("GVL") wird verwendet, um globale Variablen zu deklarieren. Wenn eine GVL im POUs-Fenster vorliegt, sind die darin enthaltenen Variablen im gesamten Projekt verfügbar. Wenn eine GVL einer bestimmten Applikation zugeordnet ist, sind die Variablen innerhalb dieser Applikation gültig.

Eine GVL kann über den Befehl Objekt hinzufügen / Globale Variablenliste hinzufügen...-Dialog ins Projekt eingefügt werden. Um sie einer bestimmten Applikation zuzuweisen, muss zunächst diese Applikation im Geräte-Fenster ausgewählt werden, ansonsten wird sie automatisch im POUs-Fenster eingefügt. Eine GVL wird durch das Symbol gekennzeichnet.

Nach dem Einfügen kann die GVL im GVL Editor bearbeitet werden.

Wenn das Zielsystem die Netzwerkfunktionalität unterstützt, können die Variablen einer GVL zu Netzwerkvariablen gemacht werden und damit für den Datenaustausch mit anderen Geräten im Netzwerk verwendet werden. Dazu müssen entsprechenden Netzwerk-Eigenschaften für die GVL definiert werden.

Zu beachten ist, dass in GVLs deklarierte Variablen ab Compiler-Version 3.2.0.0 immer vor den lokalen Variablen von POUs **initialisiert** werden.

3.7.12 Globale Netzwerkvariablenliste - GNVL

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Eine globale Netzwerkvariablenliste (Global NVL, GNVL) wird nur im Gerätebaum verwendet. Sie enthält Variablen, die in einem anderen Gerät des Netzwerks als Netzwerkvariablen definiert sind.

Das heißt: ein GNVL Objekt kann einer Applikation hinzugefügt werden, wenn mindestens eine GVL mit speziellen Netzwerkeigenschaften in einem anderen Gerät vorliegt, egal ob innerhalb desselben Projekts oder in verschiedenen. Wenn mehrere geeigneter GVLs im aktuellen Projekt für das Netzwerk gibt, kann beim Einfügen der GNVL im Objekt hinzufügen-Dialog die gewünschte aus einer Auswahlliste genommen werden.

Eine GNVL im aktuellen Gerät (Empfänger) entspricht also immer genau einer GVL in einem anderen Gerät (Sender).

Neben einem Namen muss beim Anlegen einer GNVL auch eine für die Handhabung der Netzwerkvariablen verantwortliche Task definiert werden. Alternativ zur direkten Auswahl einer SenderGVL von einem anderen Gerät kann eine Exportdatei *.gvl angegeben werden, die vorher aus dieser GVL über die "Verknüpfung mit Datei" Eigenschaften erzeugt wurde. Dies ist auf jeden Fall erforderlich, wenn die gewünschte GVL in einem anderen Projekt definiert ist. Man wählt dazu die Option "mport aus Datei" in der Sender Auswahlliste und gibt dann den Dateipfad im Eingabefeld "Import aus Datei:" ein (über Schaltfläche steht der Standarddialog für Dateiauswahl zur Verfügung).

Die Einstellungen für eine globale Netzwerkvariablenliste können später jederzeit in den Objekt-Eigenschaften (Netzwerkeinstellungen) wieder bearbeitet werden.

Sowohl die zugrundeliegende GVL als auch die zugehörige GNVL werden im Gerätebaum durch das Symbol als Netzwerkvariablenlisten gekennzeichnet.

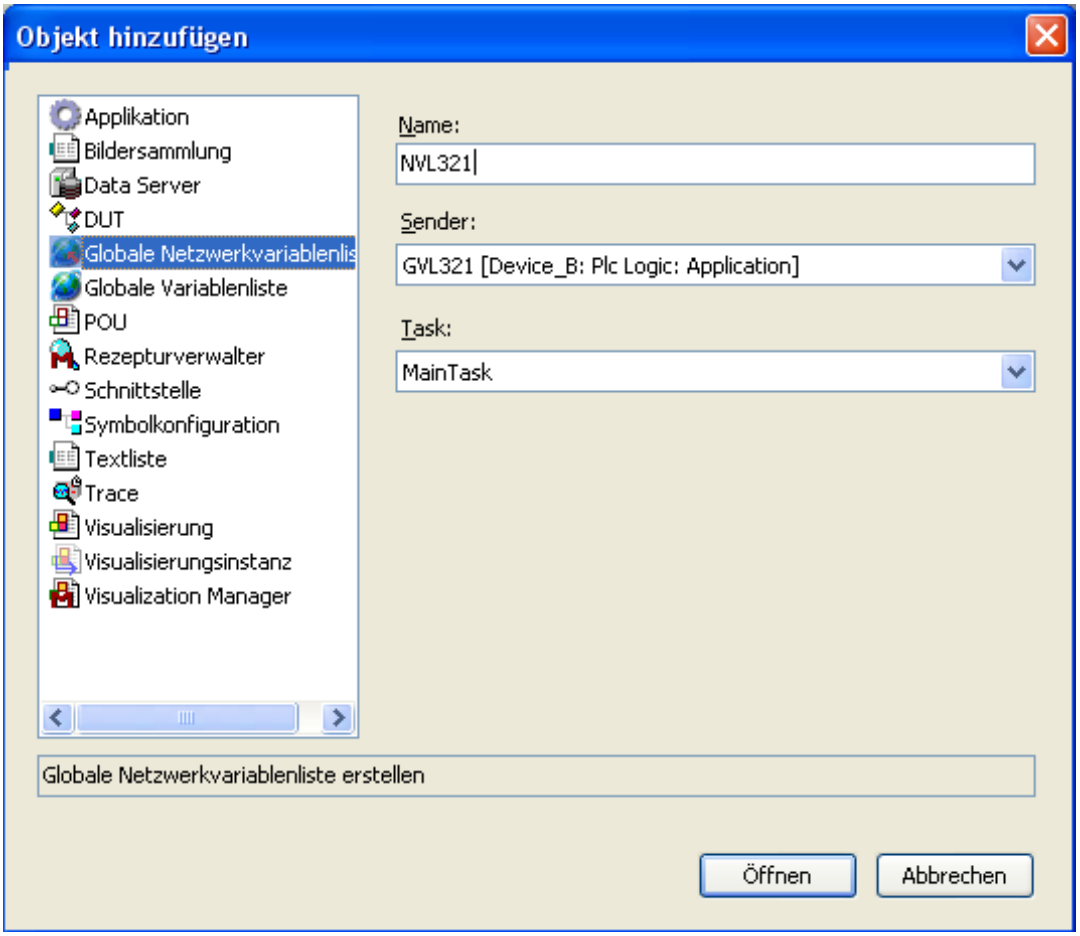


Abb. 11 Dialog 'Objekt hinzufügen, Globale NVL

Eine GNVL wird in einem Editorfenster (NVL Editor) angezeigt, der Inhalt kann jedoch vom Anwender nicht bearbeitet werden. Die Liste zeigt die gleichen Variablendeklarationen wie die zugehörige GVL, wenn die GVL verändert wird, wird die GNVL entsprechend aktualisiert.

Oberhalb der Deklarationen wird der GNVL automatisch ein Kommentar hinzugefügt, der Informationen über den Sender (Gerätepfad des Geräts, in dem die GVL liegt), den GVL-Namen und den Protokolltyp gibt.

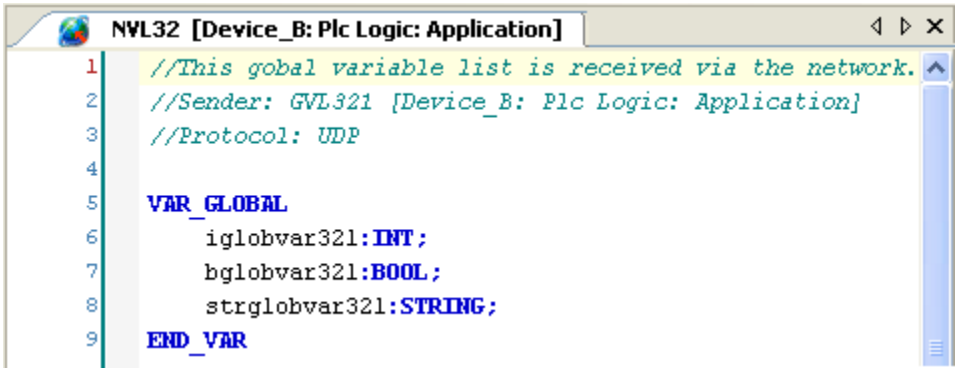


Abb. 12 Globale Netzwerkvariablenliste, Beispiel

3.7.13 Persistente Variablen

Dieses Objekt ist eine globale Variablenliste, die jedoch nur die persistenten Variablen einer Applikation enthält. Das heißt, es muss einer Applikation zugeordnet sein und dazu im Gerätebaum über den 'Objekt hinzufügen'-Dialog unterhalb dieser eingehängt werden. Das Objekt wird durch das Symbol gekennzeichnet.

Eine persistente Variablenliste wird im GVL Editor erstellt, wobei VAR_GLOBAL PERSISTENT bereits automatisch in der ersten Zeile vorgegeben ist.

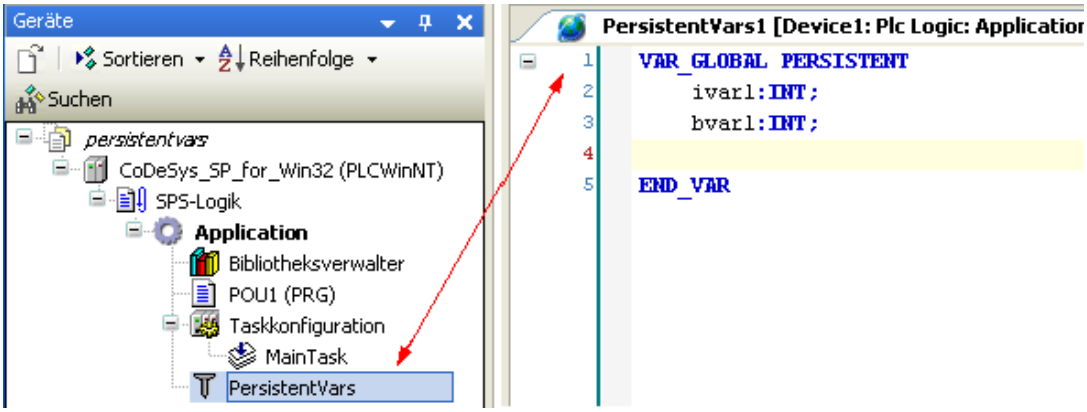


Abb. 13 Beispiel Persistente Variablenliste

Persistente Variablen werden nur bei einem Reboot oder 'Reset Ursprung <Applikation>' neu initialisiert.

3.7.14 Externe Datei

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Eine beliebige externe Datei kann über den 'Objekt hinzufügen'-Dialog im **POUs-Fenster** dem Projekt hinzugefügt werden: Wählen Sie den Typ 'Externe Datei' und geben Sie den entsprechenden **Dateipfad** an. Dazu können Sie über Schaltfläche ... den Standard-Dialog zum Öffnen einer Datei zu Hilfe nehmen. Im Eingabefeld **Name:** wird dann automatisch der Name der gewählten Datei ohne Erweiterung eingetragen. Das Feld kann jedoch editiert werden, wenn die Datei im Projekt unter einem anderen Namen verwaltet werden soll.

Wie soll die Verknüpfung zu der externen Datei gehandhabt werden ?

Wählen Sie im Dialog eine der folgenden Optionen:

Die Verknüpfung speichern: Die Datei wird nur im Projekt verfügbar sein, wenn sie im angegebenen Pfad vorliegt.

Die Verknüpfung speichern und ins Projekt einbetten: Eine Kopie der Datei wird intern im Projekt gespeichert. Zusätzlich wird die Verknüpfung zum angegebenen Pfad behalten. Solange die externe Datei dort gefunden wird, gilt die unten gewählte Update-Option.
Ansonsten wird nur die im Projekt gespeicherte Version verwendet.

Ins Projekt einbetten: Nur eine Kopie der Datei wird im Projekt gespeichert. Es gibt keine Verknüpfung mehr zum externen Pfad.

Wenn sich die externe Datei ändert:

Wenn die externe Datei mit dem Projekt verknüpft ist (s.o.), kann gewählt werden, was bei Änderungen der Datei geschehen soll:

Die Datei automatisch neu laden: Die Datei wird im Projekt aktualisiert, sobald sich die externe Datei geändert hat.

Nachfragen, ob die Datei neu geladen werden soll.: Sobald sich die externe Datei geändert hat, erscheint ein Dialog, abfragt, ob die Datei auch im Projekt aktualisiert werden soll.

Nichts tun: Die Datei im Projekt bleibt unverändert, auch wenn sich die externe Datei geändert hat.

Dateieigenschaften anzeigen... Diese Schaltfläche öffnet den Standard-Dialog für Dateieigenschaften, der auch erscheint, wenn das Datei-Objekt im POUs-Fenster selektiert ist und der Befehl 'Eigenschaften' ausgeführt wird. Er enthält u.a. das Registerblatt 'Externe Datei'. Hier können die Eigenschaften, die im 'Objekt hinzufügen-Dialog definiert wurden, angesehen und verändert werden.

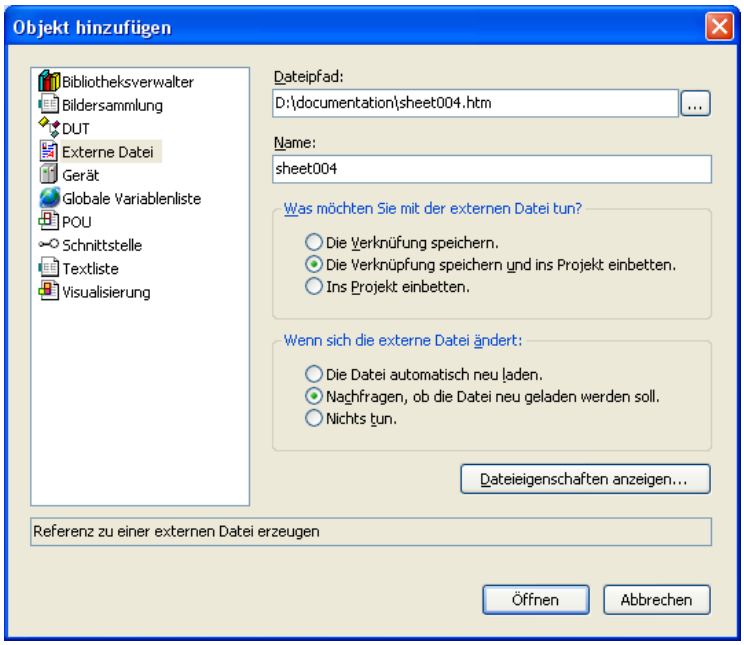


Abb. 14 'Objekt hinzufügen'-Dialog für das Hinzufügen einer externen Datei

Wenn die Einstellungen mit **Öffnen** bestätigt werden, wird das Datei-Objekt im POUs-Fenster eingefügt und die Datei im entsprechenden Editor geöffnet.

3.7.15 Textliste

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Eine Textliste wird als Objekt global im POUs-Fenster oder einer Applikation zugeordnet im Geräte-Fenster verwaltet. Sie ist die Basis für

1. **Mehrsprachigkeit (multi-language support)** für statische und dynamische Texte und Tooltips in Visualisierungen und im Alarm-Handling und für

2. **Dynamische Textwechsel**.

Textlisten können exportiert und (re-) importiert werden. **Export** ist erforderlich, wenn für eine Target-Visualisierung eine Sprachdatei in XML-Format zur Verfügung gestellt werden muss, kann aber auch für Übersetzungszwecke nützlich sein (siehe unten).

Mögliche **Formate** von Textlisten sind Textformat und XML. Die Unterstützung von **Unicode** kann aktiviert werden.

Jede Textliste ist über ihren **Namensraum** eindeutig definiert. Sie enthält Zeichenfolgen, die innerhalb der Liste durch eine **ID** (Kennzeichner, Index) und ein **Sprachenkürzel** (jeweils beliebige Zeichenfolge) eindeutig referenziert sind. Welche Textliste zu verwenden ist, wird in der Konfiguration eines Visualisierungselements festgelegt.

Der jeweilige Text wird dann im Online-Betrieb abhängig von der gerade in der Visualisierung eingestellten Landessprache dargestellt. Die verwendete Sprache kann durch eine "Sprachumschaltung"s-Eingabeaktion geändert werden, d.h. über eine entsprechend konfigurierte Mausaktion auf ein Visualisierungselement. Jeder Text ist in der Liste zumindest in der "Standard"-Sprache verfügbar (welche dies ist, Wenn es in der Textliste keinen Eintrag gibt, der mit der gerade eingestellten Landessprache übereinstimmt, wird der Standard-Sprachen-Eintrag verwendet. Jeder Text kann Formatierungsangaben enthalten.

Struktur einer Textliste:

ID (Index)	Standard	<Sprache 1>	<Sprache 2>	.... <Sprache n>
<eindeutige Zeichenfolge>	<Text abc in Standard-Sprache>	<Text abc in Sprache 1>	<Text abc in Sprache 2>	...
<eindeutige Zeichenfolge>	<Text xyz in Standard-Sprache>	<Text xyz in Sprache 1>	<Text xyz in Sprache 2>	...

Textlistentypen

Es gibt zwei Typen von Text, die in Visualisierungselementen verwendet werden können und entsprechend gibt es zwei Typen von Textlisten:

1. GlobalTextList **für statische Texte:**

Statische Texte in einer Visualisierung können im Gegensatz zu dynamischen **nicht über eine Variable im Online-Modus ausgetauscht** werden. Nur die Version der Landessprache wird angepaßt, wenn sie durch einen "Sprachumschaltung"s-Eingabeaktion auf ein Visualisierungselement umgeschaltet wird. Ein statischer Text wird einem Visualisierungselement über die Eigenschaft 'Text' oder 'Tooltip' der Kategorie 'Texts' zugewiesen. Sobald der erste statische Text in einem Projekt definiert wird, wird eine Textliste mit Namen "GlobalTextList" **automatisch** als Objekt im POUs-Fenster angelegt. Zunächst enthält die Liste den definierten Text in Spalte "Standard" und eine automatisch vergebene ganze Zahl (beginnend mit 0) als Text-"ID". Weitere statische Texte werden dann hinzugefügt, sobald sie in den Eigenschaften eines Visualisierungselements definiert werden. Die ID-Nummer wird dabei jeweils um 1 hochgezählt.

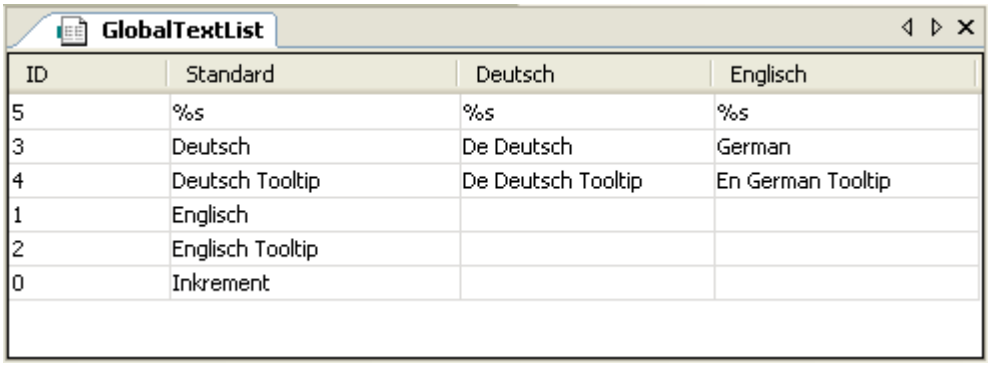
Wird innerhalb eines Visualisierungselements ein statischer Text eingegeben (z.B. unter Rechteck „Texts“ „Text“ wird der Text „Beispiel“ eingegeben), so wird innerhalb der GlobalTextList nach diesem Text gesucht.

- Wird der Text gefunden (z.B. ID „4711“, Text „Beispiel“) so wird in dem Element in einer internen Variablen „TextId“ der Wert 4711 eingetragen. Somit wird eine Verbindung zwischen dem Element und der Zeile innerhalb der GlobalTextList geschaffen.
- Wird der Text nicht gefunden so wird eine neue Zeile innerhalb der GlobalTextList eingetragen (z.B. ID „4712“, Text „Beispiel“). In dem Element wird nun in der internen Variablen TextId der Wert 4712 aufgenommen.

Dies bedeutet, dass bei jeder Änderung eines statischen Textes innerhalb der Visualisierung evtl. auch eine Änderung innerhalb der GlobalTextList erfolgt.

Wenn es sie noch nicht gibt, kann eine globale Textliste auch **explizit angelegt** werden: Befehl Globale Textliste anlegen.

GlobalTextList ist eine spezielle Textliste, in der die Kennzeichner (IDs) für die einzelnen Texteinträge implizit gehandhabt werden und vom Anwender im PLC Designer nicht editiert werden können. Auch kann diese Liste nicht gelöscht werden. Aber sie kann exportiert, extern bearbeitet und dann re-importiert werden. In diesem Fall wird beim Re-Import geprüft, ob die Kennzeichner noch mit denen übereinstimmen, die in der Konfiguration des jeweiligen Visualisierungselements angegeben sind. Wenn nötig, wird eine implizite Aktualisierung der IDs in der Element-Konfiguration durchgeführt.



GlobalTextList

ID	Standard	Deutsch	Englisch
5	%s	%s	%s
3	Deutsch	De Deutsch	German
4	Deutsch Tooltip	De Deutsch Tooltip	En German Tooltip
1	Englisch		
2	Englisch Tooltip		
0	Inkrement		

Abb. 15 Beispiel von GlobalTextList

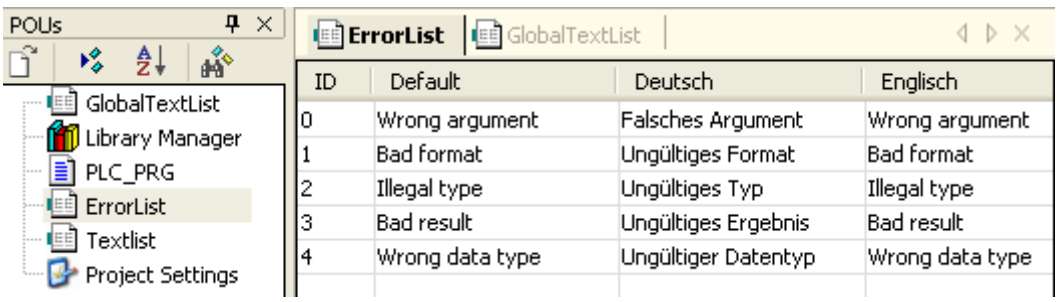
2. Textliste für dynamische Texte**:**

Dynamische Texte können **dynamisch im Online-Betrieb ausgetauscht** werden (s.o.). Der Text-Index (**ID)**, eine Zeichenfolge, muss innerhalb der verwendeten Textliste eindeutig sein und muss im Gegensatz zu GlobalTextList vom Anwender vergeben werden. Ebenfalls anders als bei GlobalTextList **müssen** Textlisten für dynamische Texte **explizit angelegt** werden ('Objekt hinzufügen'-Dialog).

Alle im Projekt verfügbaren dynamischen Textlisten werden angeboten, wenn man die Eigenschaft 'Dynamische Texte' / 'Textliste' eines Visualisierungselements konfiguriert. Wenn man dort den Namen einer Textliste und zusätzlich einen Text-Index (ID) aus der Liste angibt, wird im Online-Betrieb der entsprechende Text angezeigt. Wenn die ID nicht absolut, sondern über eine Projektvariable eingegeben wird, kann der Text über diese Variable dynamisch umgeschaltet werden.

Eine dynamische Textliste **muss exportiert werden, wenn** sie als Sprachdatei für die Sprachumschaltung in der Target-Visualisierung verwendet werden soll. Der Dateipfad muss in den Visualisierungsoptionen angegeben werden. Wie GlobalTextList kann eine dynamische Textliste auch zum Zweck des externen Bearbeitens exportiert und dann re-importiert werden.

Hinweis: Im Gegensatz zu GlobalTextList gibt es beim Re-Importieren von dynamischen Textlisten **keine automatische Überprüfung und Aktualisierung der IDs!** Stellen Sie also sicher, dass die Index-IDs beim Bearbeiten von exportierten Listen nicht verändert werden!



ID	Default	Deutsch	Englisch
0	Wrong argument	Falsches Argument	Wrong argument
1	Bad format	Ungültiges Format	Bad format
2	Illegal type	Ungültiges Typ	Illegal type
3	Bad result	Ungültiges Ergebnis	Bad result
4	Wrong data type	Ungültiger Datentyp	Wrong data type

Abb. 16 Beispiel einer dynamischen Textliste mit Namen 'ErrorList'

Beschreibung eines Beispiels

Angenommen Sie wollen ein Visualisierungselement konfigurieren, das später immer beim Aufttreten eines Fehlers die entsprechende Fehlermeldung ausgeben soll.

Die Applikation verarbeitet Fehler, die über numerische IDs - zugewiesen einer Integer-Variable *ivar_err* - gekennzeichnet sind.

Stellen Sie eine dynamische Textliste namens 'ErrorList' bereit, wobei die Fehlermeldungstexte für Fehler IDs "0" bis "4" in den Sprachen 'Deutsch', 'Englisch' und 'Default' definiert sind:

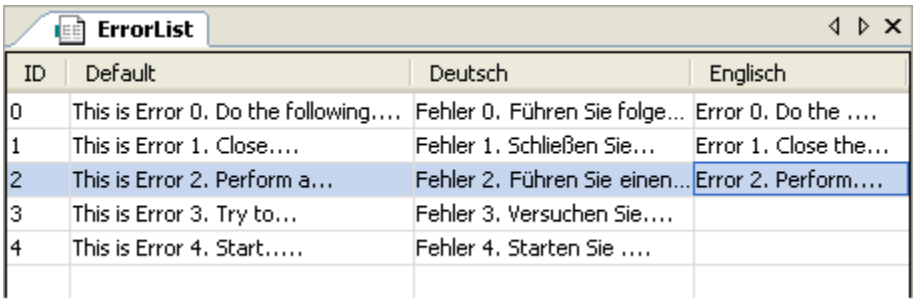


Abb. 17 Example 'ErrorList'

Für die Verwendung der Fehler-IDs in einer Visualisierungskonfiguration deklarieren Sie eine STRING-Variable, z.B. *strvar_err*. Um den Wert von *ivar_err* der Variablen *strvar_err* zuzuweisen, verwenden Sie strvar_err:=INT_TO_STRING(ivar_err);)

strvar_err kann nun als 'Textindex' parameter in der Konfiguration der 'Dynamic texts' Eigenschaften eines Visualisierungselements verwendet werden. Dann wird das Element im Online-Betrieb die entsprechende Fehlermeldung anzeigen.

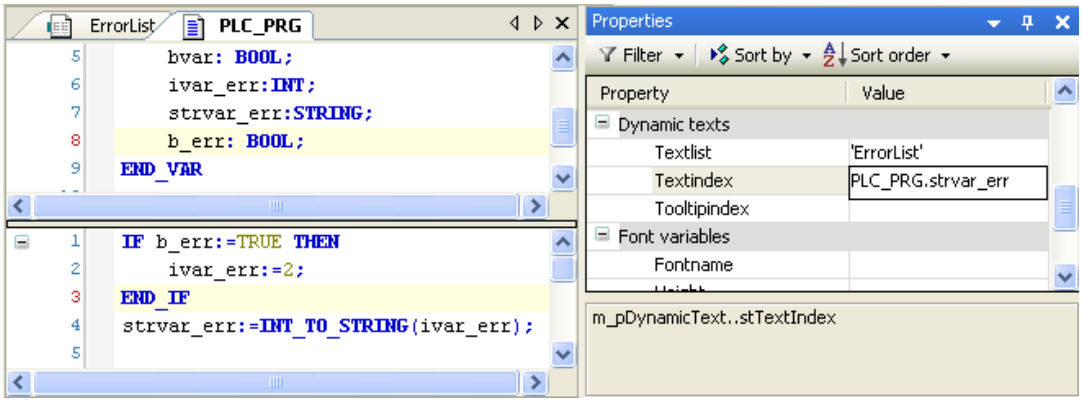


Abb. 18 Beispiel: Projektvariablen, die die Fehler-ID verarbeiten, Konfiguration eines Visualisieurngselements (Eigenschaften-Dialog), das die Fehlermeldungen ausgeben soll.

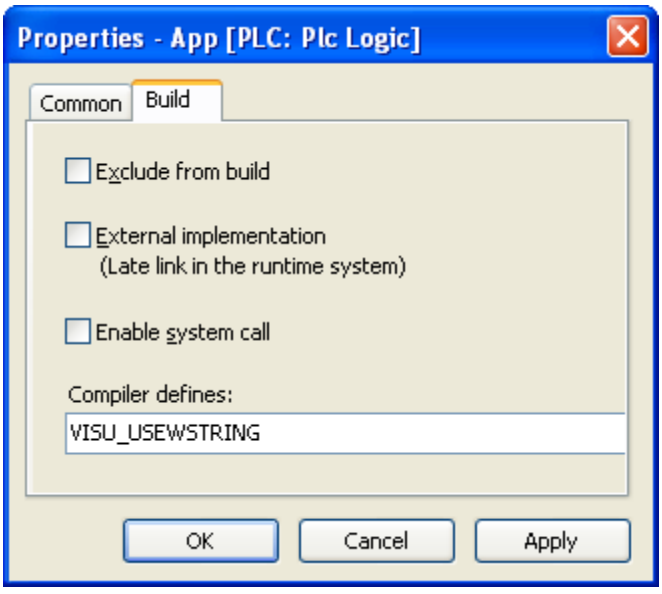
Textliste anlegen

- Zum Anlegen einer Textliste für dynamische Texte fügen Sie mit Hilfe des Befehls 'Objekt hinzufügen' ein 'Textliste'-Objekt im POUs-Fenster oder unterhalb der gerade ausgewählten Applikation im Geräte-Fenster ein. Dabei geben Sie einen Namen für die Liste an und mit diesem Namen wird die Liste im POUs oder Geräte-Baum eingefügt und gleichzeitig im Textlisten-Editor geöffnet.

- Um die Textliste für statische Texte 'GlobalTextList' anzulegen, tragen Sie entweder beim Konfigurieren eines Visualisierungselements in der Eigenschaften-Kategorie 'Texte' bei Eigenschaft 'Text' einen Text ein, was zum automatischen Erzeugen der Liste führt, oder sie generieren sie explizit über den Befehl 'Globale Textliste anlegen'.
- Um eine bestehende Textliste zum Bearbeiten zu öffnen, verwenden Sie den Befehl 'Objekt bearbeiten' oder führen Sie einen Doppel-Klick auf den Objekteintrag aus.
- Um ein Feld in der Textliste zu editieren, wählen Sie das Feld erst mit einem Mausklick an und führen dann einen weiteren Mausklick oder die <Leertaste> aus, um ein Eingabefeld zu öffnen. Geben Sie die gewünschte Zeichenfolge ein und schließen das Feld mit <Eingabe>-Taste. MIt den Pfeiltasten gelangen Sie zum nächsten/vorherigen Feld.

Unicode-Format Unterstützung

Wenn das Unicode-Format unterstützt werden soll, muss die entsprechende Option im Visualization Manager aktiviert werden. Außerdem muss eine spezielle Compiler-Anweisung für die Applikation eingegeben werden. Dazu wird die Applikation im Gerätebaum angewählt und der 'Eigenschaften'-Dialog, Kategorie 'Übersetzen' geöffnet. Im Feld Compiler- Definitionen geben Sie ein: "VISU_USEWSTRING".



Beachten Sie auch den nachfolgenden Absatz für den Export von Textlisten.

Export und Import von Textlisten

Statische und dynamische Textlisten können in Text- oder XML-Format exportiert werden. Wenn eine Sprachdatei in "normalem" oder in Unicode-unterstützendem Format für die Target-Visualisierung benötigt wird, muss diese explizit durch einen Unicode-Export der betreffenden Textliste erzeugt werden. Exportierte Dateien können auch verwendet werden, um extern Texte hinzuzufügen, z.B. durch einen Übersetzer. Beachten Sie jedoch, dass **nur Dateien im Textformat (*.txt, *.csv) auch wieder importiert** werden können.

Sehen Sie die ausführlichen Beschreibungen der entsprechenden Befehle unter: Befehle der Kategorie Textlisten.

Das Verzeichnis, in dem die Export-Dateien gespeichert werden sollen, ist in den Visualisierungs-Projekteinstellungen definiert.

Formatierung von Texten

Die Texte können Formatierungsangaben enthalten (%s,%d,...), die es u.a. ermöglichen, aktuelle Variablenwerte im Text wiederzugeben. Sehen Sie die Hilfeseite zu Text in Visualisierungen die erlaubten Formatierungsdefinitionen.

Die Textdefinition wird in folgender Reihenfolge ausgewertet, um den Text jeweils aktuell darzustellen:

1. Der zu verwendende Text wird über Listenname und Text ID ermittelt.

2. Wenn der Text Formatierungsangaben enthält, werden diese durch den Wert der entsprechenden Variable ersetzt.

Nachlieferung von Übersetzungen

Durch das Einfügen der GlobalTextList.csv (Nachlieferungsdatei) in das Verzeichnis welches für das Laden der Textlisten verwendet wird kann eine Nachlieferung von Übersetzungen erfolgen. Beim Starten des Bootprojekts wird festgestellt das eine Nachlieferungsdatei existiert und es werden die Übersetzungen mit denen der Textlistdateien abgeglichen. Dabei werden neue und auch geänderte Übersetzungen in die Textlistdateien übernommen.

Danach wird die Datei GlobalTextList.csv als geladen markiert. Eine Nachlieferung von Texten wirkt sich somit nur einmal auf die Startup-Zeit des Bootprojektes aus.

"Komponenten Auflisten" für die Texteingabe

Über die Visualisierungsoptionen kann eine Textvorlagedatei angegeben werden. Alle Texte der Spalte „Default“ dieser Datei werden in eine Liste aufgenommen welche für die "Komponenten Auflisten“ Funktion herangezogen werden. Als Textvorlagedatei kann eine Datei verwendet werden, die zuvor über das Export-Kommando erzeugt wurde.

Mehrbenutzerbetrieb

Durch die Nutzung der Quellcodeverwaltung in PLC Designer ist es möglich dass mehrere Benutzer gleichzeitig an einem Projekt arbeiten. Dabei müssen die nachfolgenden Punkte beachtet werden:

- Wird ein statischer Text innerhalb eines Visualisierungselements verändert, so muss die Visualisierung und evtl. auch die GlobalTextList das Schreibrecht besitzen (siehe GlobalTextList). Besitzt die GlobalTextList das Schreibrecht nicht, so sollten auch keine Texte in Visualisierungen verändert werden. Wird dies trotzdem durchgeführt, so kann es vorkommen dass die Text-Ids nicht mehr zu den Texten innerhalb eines Visualisierungselements passen.

- Durch das Kommando „Visualisierungs-Text-Ids prüfen“ können solche Fehler innerhalb aller Visualisierungen ermittelt werden.
- Durch das Kommando „Visualisierungs-Text-Ids aktualisieren“ können diese Fehlerfälle automatisch behoben werden. Dazu müssen alle Visualisierungen mit Fehlerfällen und die GlobalTextList das Schreibrecht besitzen.

Eine Auslieferung mit Fehlerfällen kann dazu führen, dass bei einer Sprachumschaltung innerhalb der Visualisierung falsche Texte erscheinen. Werden zu einem Projekt keine Fehlerfälle gemeldet, so kann die Sprachdatei auch nachträglich übersetzt und nachgeliefert werden.

Verwendung von Textlisten für die Sprachumschaltung in Visualisierungen

Wenn eine entsprechende Textliste bereitsteht, d.h. eine Textliste, die mehrere Sprachversionen für einen Text definiert, dann kann die Sprache, in der die Texte innerhalb der Visualisierung erscheinen, im Online-Betrieb durch eine Eingabe auf ein Visualisierungselement umgeschaltet werden. Dazu müssen die Elementeigenschaften "Dynamic Texts" des Elements die zu verwendende Textliste angeben und eine "OnMouse.." Eingabeaktion "Sprachumschaltung" muss konfiguriert sein, die die Sprache angibt, auf die durch die Mausaktion umgeschaltet werden soll. Die Sprache muss dabei genau so angegeben werden, wie sie in der Spaltenüberschrift der zugehörigen Textliste steht.

3.7.16 Bildersammlung

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Bildersammlungen sind Tabellen, die den Pfad, eine Vorschau und einen Kennzeichner (ID, Zeichenfolge) für jede Abbildungsdatei definieren. Durch Angabe der ID und - für eindeutiges Ansprechen - zusätzlich des Namens der Bildersammlung, kann eine Bilddatei referenziert werden, wenn sie im Projekt beispielsweise in einer Visualisierung verwendet wird (in der Konfiguration der Eigenschaften eines Visuailsierungselements, siehe unten).

Hinweis: Es wird empfohlen die Größe einer Bilddatei vor ihrer Einbindung so weit wie möglich zu reduzieren. Anderenfalls führt die Größe des Projekts zu unverhältnismäßig langen Lade- und Speicherzeiten bei Visualisierungen oder Bildeinbindungen.

Struktur einer Bildersammlung:

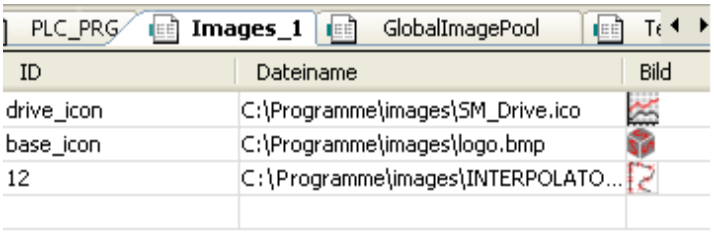


Abb. 19 Beispiel einer Bildersammlung

ID: Kennzeichner in Form einer Zeichenfolge, z.B. "logo", "base_icon", "2"); eine eindeutige Referenzierung einer Bilddatei wird über die Kombination des Namens der Bildersammlung mit der ID der Bilddatei (z.B. "List1.basic_logo") erreicht.

Dateiname: Pfad der Bilddatei (z.B. "C:\Programme\images\logo.bmp")

Bild: Vorschau der Abbildung

Anlegen und Bearbeiten einer Bildersammlung

Ein Projekt kann mehrere Bildersammlungen enthalten.

Wenn im Projekt noch keine Bildersammlung vorliegt, wird - sobald das erste "Image"-Element in die Visualisierung eingefügt und eine ID (static ID) in den Elementeigenschaften eingetragen wird - automatisch eine Bildersammlung mit dem Standardnamen "**GlobalImagePool**" und einem Eintrag für die gewählte Bilddatei angelegt. GlobalImagePool ist eine globale Sammlung, die immer als erste durchsucht wird, wenn eine Bilddatei zur Verwendung ansteht. Zusätzlich können individuell benannte Sammlungen verwendet werden.

Manuell können Bildersammlungen folgendermaßen angelegt werden: GlobalImagePool über den Befehl Globale Bildersammlung erzeugen, die anderen über den Befehl Objekt hinzufügen. Sie werden im POUs Fenster oder direkt unterhalb einer Applikation im Geräte-Fenster eingefügt. Dazu wird im 'Objekt hinzufügen'-Dialog der Objekttyp 'Bildersammlung' ausgewählt und ein Name für die Sammlung vergeben.

Um eine Bilddatei manuell einer Bildersammlung hinzuzufügen, setzen Sie den Fokus in das **ID-**Feld der ersten leeren Zeile in der Tabelle der Sammlung, drücken die <Leer>-Taste, um ein Eingabefeld zu öffnen und geben eine beliebige Zeichenfolge als ID ein. Wenn die angegebene ID innerhalb der Tabelle schon verwendet wird, wird automatisch eine Ziffer angehängt, beginnend mit 0 und hochgezählt um 1für jede weitere Kopie der ID.

Setzen Sie anschließend den Cursor in das Feld **Dateiname**. Hier kann über die <Leer>-Taste und dann über die Schaltfläche [...] der **Bildauswahl-**Dialog geöffnet werden, über den der Pfad der gewünschten Bilddatei eingegeben wird.

Verwendung von Bilddateien aus Bildersammlungen

Beachten Sie folgendes, wenn die ID einer Bilddatei in mehreren Bildersammlungen vorkommt:

- **Suchreihenfolge**: Wenn eine Abbildung gewählt wird, die im GlobalImagePool verwaltet wird, muss der Name der Bildsammlung nicht angegeben werden. Die Suchreihenfolge für Abbildungsdateien entspricht übertragen der für globale Variablen: 1. globaler GlobalImagePool, 2. Bildersammlungen, die der gerade aktiven Applikation zugewiesen sind, 3. Bildersammlungen, die neben GlobalImagePool im POUs-Fenster liegen, 4. Bildersammlungen in Bibliotheken.
- **Eindeutiger Zugriff**: Sie können die gewünschte Bilddatei direkt und eindeutig ansprechen, indem der ID der Name der Bildersammlung vorangestellt wird, gemäß Syntax "<Name der Sammlung>.<Bild ID>" (z.B. für das oben in der Abbildung gezeigte Beispiel: "Images_1.drive_icon>").

1. Verwenden einer Bilddatei in einem Visualisierungselement vom Typ "Image":

Wenn ein Image-Element in einer Visualisierung eingefügt wird, kann festgelegt werden, ob es ein statisches Element sein soll oder ein dynamisches, welches im Online-Betrieb in Abhängigkeit von einer Variable ausgetauscht werden kann:

Statische Abbildungen:

In der Konfiguration des Elements (Eigenschaft "**Static ID**") wird die ID der Bilddatei bzw. der Name der Bildersammlung + ID eingegeben. Beachten Sie hierzu die Bemerkungen zur Suchreihenfolge und Eindeutigem Zugriff (s.o.).

Dynamische Abbildungen:

In der Konfiguration des Elements (Eigenschaft "**Bitmap ID variable**") wird die Variable eingegeben, die die ID der Bilddatei definiert, z.B. "PLC_PRG.imagevar".

2. Verwenden einer Bilddatei für den Visualisierungshintergrund:

In der Hintergrund-Definition für eine Visualisierung kann eine Abbildung angegeben werden. Die Bilddatei kann wie oben für ein Visualisierungselement beschrieben durch den Namen der Bildersammlung und den Dateinamen angegeben werden.

3.7.17 Visualisierung

Information zur Visualisierung finden Sie im Abschnitt Visualisierungs-Editor.

3.7.18 POUs für implizite Prüfungen

Diese speziellen POUs können einer Applikation hinzugefügt werden, um sie mit implizit zur Verfügung gestellten Überwachungsfunktionalitäten auszustatten. Diese überprüfen während der Laufzeit die Grenzen von Arrays oder Unterbereichstypen, die Gültigkeit von Pointeradressen oder eine Division durch 0.

Zu diesem Zweck stehen im Dialog "Objekt hinzufügen" in der Kategorie "POUs für implizite Prüfungen" folgende Funktionen zur Verfügung:

- CheckBounds
- CheckDivInt
- CheckDivLInt
- CheckDivReal
- CheckDivLreal
- CheckRange
- CheckRangeUnsigned
- CheckPointer

Nach dem Einfügen einer POU zur Überwachung wird diese im zur gewählten Implementierungssprache passenden Editor geöffnet. Ein Vorschlag zur Implementierung ist im ST Editor gemacht und kann beliebig angepasst werden.

Eine bereits eingefügte Überwachungsfunktion steht im Dialog "Objekt einfügen" nicht länger zur Auswahl um ein mehrfaches Einbinden zu verhindern. Falls bereits alle Typen von Überwachungsfunktionen eingefügt wurden, steht im Dialog die Kategorie "POUs für implizite Prüfungen" insgesamt nicht mehr zur Auswahl.

Achtung: Um die Funktionalität der Überwachungsfunktionen zu erhalten, darf der Deklarationsteil nicht verändert werden.

3.8 Gerät, Gerätebaum

Im Geräte-Fenster ("**Gerätebaum**") kann die Hardware abgebildet werden, auf der die Applikation(en) laufen soll(en).

- Jedes "**Gerät**"-Objekt repräsentiert eine bestimmte (Zielsystem) Hardware. Beispiele: Steuerungsgerät, Feldbusknoten, Buskoppler, Antrieb, E/A-Modul, Monitor.
- Jedes Gerät ist durch seine Gerätebeschreibung definiert und muss auf dem lokalen System installiert sein, um für das Einfügen im Gerätebaum verfügbar zu sein. Die Gerätebeschreibungsdatei definiert die Eigenschaften eines Geräts hinsichtlich Konfigurierbarkeit, Programmierbarkeit und möglichen Verbindungen zu anderen Geräten.
- Im Gerätebaum werden jedoch nicht nur die reinen Geräte-Objekte verwaltet, sondern alle Objekte, die benötigt werden, um eine Applikation (also ein Steuerungsprogramm) auf einem Gerät (Steuerung, SPS) auszuführen; also auch die 'Applikation"-Objekte, sowie "Taskkonfiguration" und "Task". Aber auch reine Programmierobjekte wie einzelne POUs, Globale Variablenlisten und Bibliotheksverwalter können hier - anstatt als projektweit instanzierbare Einheiten im POUs-Fenster - NUR im Gerätebaum verwaltet werden und sind dann eben nur für "ihre" Applikation bzw. deren "Kind-Applikationen" verwendbar.

Beachten Sie die Möglichkeit die aktive Applikation auf einem "**Simulationsgerät**" laufen zu lassen, das standardmäßig innerhalb des Programmiersystems automatisch zur Verfügung steht. So wird kein echtes Zielgerät benötigt um das Online-Verhalten der Applikation zu testen. Sobald Sie in den Simulationsmodus wechseln, wird der Eintrag PLC Designer SP Win V3 im Gerätebaum in Kursivdruck dargestellt; die Applikation kann eingeloggt werden.

Sehen Sie im folgenden Generelle Informationen zum Gerätebaum und Hinweise zur Installation und dem Anordnen der Objekte.

Zur Konvertierung von Geräte-Referenzen beim Öffnen von Projekten, die in einem anderen Format erstellt wurden, sehen Sie bitte unter Projekt öffnen.

Generelles:

- Der **Wurzelknoten** des Baums ist immer ein symbolischer Knoteneintrag: *<Projektname>*.
- Die Konfigurationsbäume der "**Steuerungskonfiguration**" und "**Taskkonfiguration**", die im PLC Designer V2.3 in separaten Fenstern gehandhabt wurden, sind im Gerätebaum integriert. Nur die Konfiguration der diversen Geräte- bzw. Task-Parameter wird in separaten Editor-Dialogen durchgeführt. Siehe hierzu: Taskkonfiguration, Steuerungskonfiguration.
  So wird die Struktur der zu steuernden Hardware-Umgebung im Gerätebaum durch die entsprechende Anordnung von Objekten abgebildet, und es ist möglich, ein heterogenes, hierarchisches System von mehrfach vernetzten Steuerungen und zugrundeliegenden Feldbussen aufzusetzen.
- Sehen Sie unten die "Regeln" für das Anordnen von Objekten im Gerätebaum.

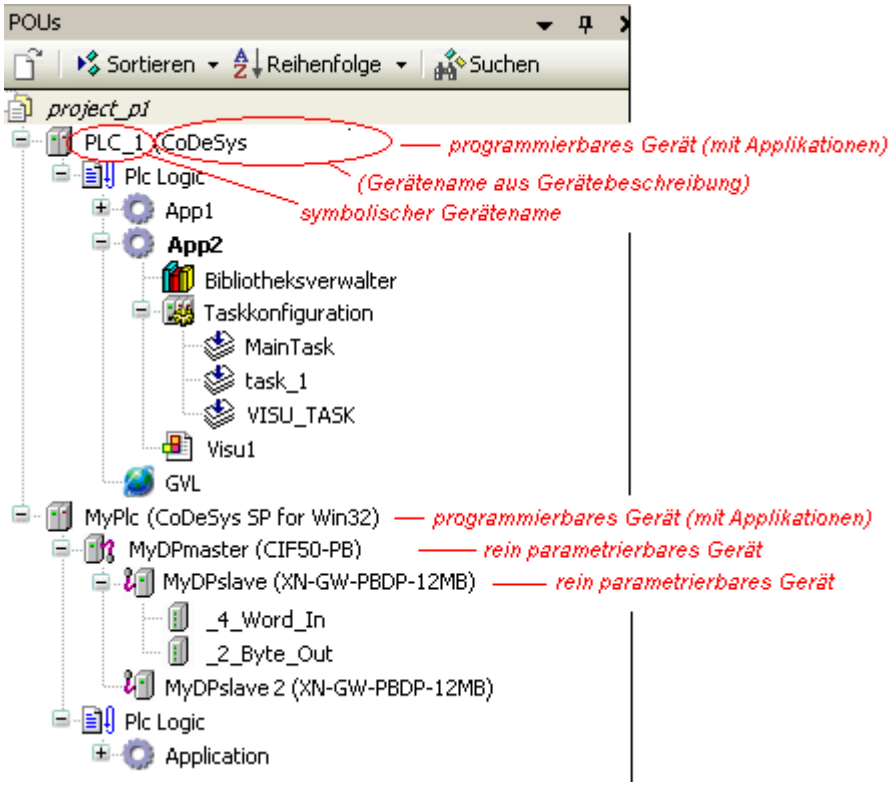


Abb. 20 Gerätebaum, Beispiel

- Ein Geräte-Eintrag im Gerätebaum zeigt jeweils das Symbol, den symbolischen (im Baum editierbaren) **Gerätenamen** und dahinter den **Gerätetyp** (=Gerätename, wie in der Gerätebeschreibung definiert)

- Es gibt **programmierbare** Geräte und nur **parametrierbare** Geräte. Der Gerätetyp bestimmt die mögliche Einfügeposition im Gerätebaum und die Auswahl der Objekte, die unterhalb des Geräts eingefügt werden können. Programmierbare Geräte erhalten automatisch einen zusätzlichen "**SPS Logik**" Knoten unterhalb des Geräte-Objekts. Unter diesem Knoten können dann die Objekte, die für das Programmieren des Geräts benötigt werden, eingefügt werden (Applikationen, GVLs, Textlisten etc.).
- Innerhalb eines Projekts können ein oder mehrere programmierbare Geräte - unabhängig von Hersteller oder Typ - konfiguriert werden ("**Multi-Ressource**", "**Multi-Device**", Netzwerke).
- Die **Konfiguration** eines Geräts bezüglich Kommunikation, Parameter, I/O-Abbild wird im **Geräte-Editor Dialog** durchgeführt, der mit einem Doppelklick auf den Geräte-Eintrag geöffnet wird (für eine Beschreibung siehe Geräte-Editor).
- **Im Online-Modus** zeigt ein Symbol vor einem Geräte-Eintrag an, ob das Gerät gerade verbunden ist oder nicht. Zusätzliche Diagnose-Information gibt der jeweilige Geräte-Dialog in Kategorie "Status".

Installieren von Geräten auf dem lokalen System

- **Installation** und De-Installation von Geräten kann im Geräte-Repository Dialog durchgeführt werden. Die Installation basiert auf **Gerätebeschreibungsdateien** im XML-Format. Die Standard-Erweiterung für eine gültige Beschreibungsdatei ist ***.devdesc.xml**. Es können aber auch busspezifische Beschreibungsdateien wie *.gsd-Dateien (Profibus), *.eds-Dateien (CANbus, ASi) etc., über den Geräte-Repository Dialog installiert werden. Die spezifischen PLC Designer V2.3-Konfigurationsdateien ***.cfg** können verwendet werden, wenn eine entsprechende Info-Datei ***.info.cfg** vorliegt, die das jeweilige Set an Konfigurationsdateien beschreibt.

Anordnen und Konfigurieren von Objekten im Gerätebaum, Vorgehen und Regeln:

- Um ein **Objekt einzufügen**, verwenden Sie den Befehl Objekt hinzufügen, um ein "Gerät" einzufügen am besten gleich den Befehl Gerät einfügen (bzw. anhängen). Welche Objekttypen eingefügt werden können, hängt von der gerade ausgewählten Position im Baum ab. Beispiel: Module für einen DP Profibus Slave können nicht eingefügt werden, ohne das entsprechende Slave-Objekt vorher eingefügt zu haben; Applikationen können nicht unterhalb von nicht-programmierbaren Geräten eingefügt werden. Weiterhin stehen immer nur solche Geräte zum Einfügen zur Auswahl, die korrekt auf dem lokalen System installiert wurden und zur gerade ausgewählten Position im Baum passen.

- Eine **Veränderung der Anordnung** ist möglich über die Standardbefehle der Kategorie "Zwischenablage" (Ausschneiden, Kopieren, Einfügen, Löschen) oder durch Ziehen des ausgewählten Objekts mit der Maus bei gedrückter Maustaste (für Kopieren plus <Strg>). Beachten Sie zum Einfügen-Befehl: Falls das ausgeschnittene Objekt unterhalb oder oberhalb des gerade selektierten Objekts eingefügt werden könnte, öffnet der '**Einfügeposition auswählen**'-Dialog, wo sie entsprechend entscheiden.
- Ein bereits im Gerätebaum eingefügtes Gerät kann durch eine andere Version desselben Geräts oder durch ein Gerät anderen Typs ersetzt werden (**Geräte-Update**). Ein eingerückt unterhalb des Geräts vorhandener Konfigurationsbaum wird dabei soweit als möglich erhalten.
- Auf der Ebene direkt unterhalb des **Wurzelknotens** *<Projektname>* können nur "Gerät"-Objekte eingehängt werden. Wenn ein anderer Objekttyp im 'Objekt hinzufügen'-Dialog ausgewählt wird, wie z.B. "Textliste", wird dieses automatisch im POUs-Fenster eingefügt!. Wenn gerade kein Eintrag selektiert ist, z.B. wenn Sie mit der Maus in den leeren Bereich des Geräte-Fensters klicken, ist dies gleichbedeutend mit dem Selektieren des Wurzelknotens.
- Ein Gerät wird als Knoten im Baum eingefügt. Wenn solche in der Gerätebeschreibung definiert sind, werden automatisch die **Unterknoten** mit eingefügt. Ein Unterknoten kann auch wiederum ein programmierbares Gerät sein.
- **Unterhalb eines "Gerät"-Objekts** können weitere Geräte eingefügt werden, die auf dem lokalen System installiert sind und im 'Objekt hinzufügen'- bzw. 'Gerät einfügen'-Dialog angeboten werden. Die **Reihenfolge** der Geräteobjekte innerhalb des Baums von oben nach unten: Pro Ebene werden immer zuerst die programmierbaren Geräte (SPS Logik) angeordnet, dann die restlichen Typen, jeweils alphabetisch sortiert.
- Eine "**Applikation**" kann nur unterhalb eines "SPS Logik- Knotens (symbolischer Knoten für programmierbare Geräte) eingefügt werden. Pro Gerät müssen alle Applikationen eindeutig benannt sein. Unterhalb jeder Applikation können dann die weiteren für die Programmierung benötigten Objekte eingehängt werden, wie z.B. Taskkonfiguration, DUT, GVL, Visualisierungen. Eine Taskkonfiguration **muss** unterhalb jeder Applikation eingefügt werden und darin die entsprechenden Programmaufrufe (POU-"Instanzen" aus dem Geräte-Fenster bzw. applikationsspezifische POUs) definiert werden. Wenn mehrere Applikationen direkt unterhalb eines Geräts hängen, ist beim I/O-Mapping des Geräts festzulegen, die Variablen welcher dieser Applikationen aktuell für die Kommunikation mit dem Zielsystem verwendet werden sollen.

- **Hierarchische Anordnung von Applikationen, Gültigkeitsbereiche**:
  Eine Applikation kann auch unterhalb einer anderen Applikation eingehängt werden: Nennen wir das entstehende Konstrukt "Vaterapplikation" - "Kindapplikation". In diesem Fall gilt folgendes: Die Kindapplikation kann auf die Objekte der Vaterapplikation zugreifen, nicht aber umgekehrt! Hintergrund: Eine Kindapplikation soll immer entfernt bzw. ausgetauscht werden können, ohne die Vaterapplikation zu beeinträchtigen. Sehen Sie dazu folgende Abbildung: 'Application' ist Vaterapplikation für 'App1', diese wiederum für 'App2':

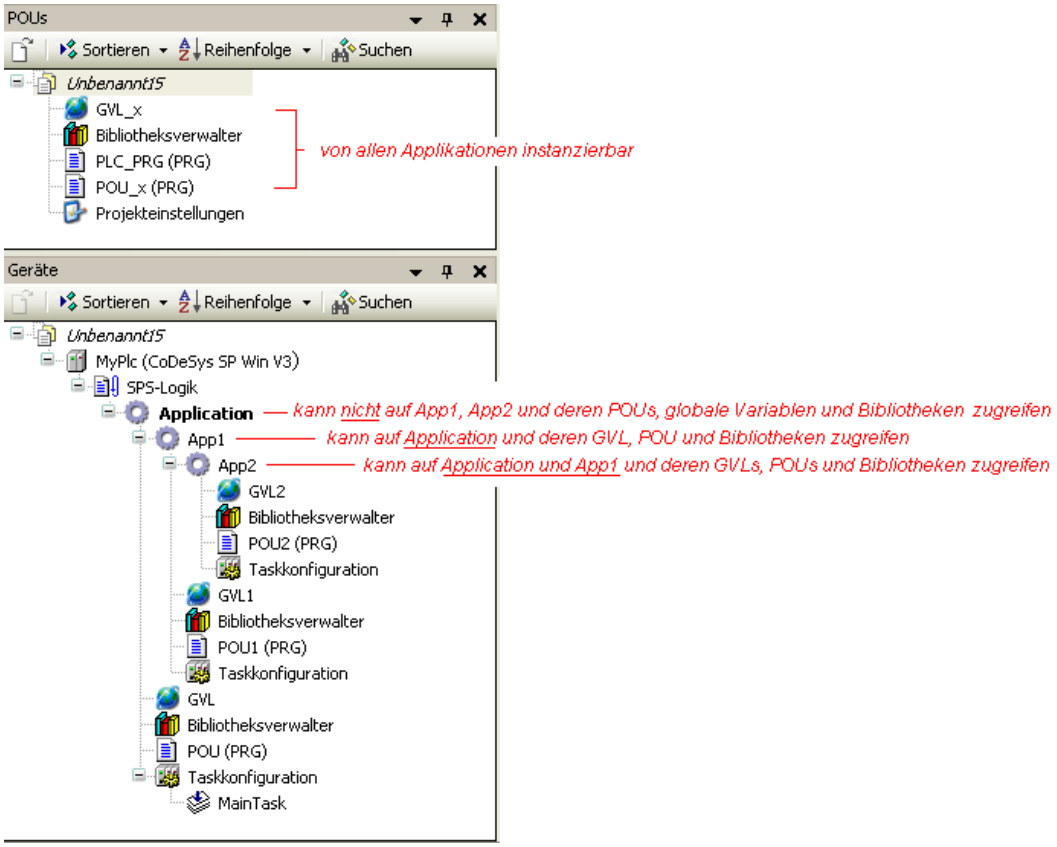
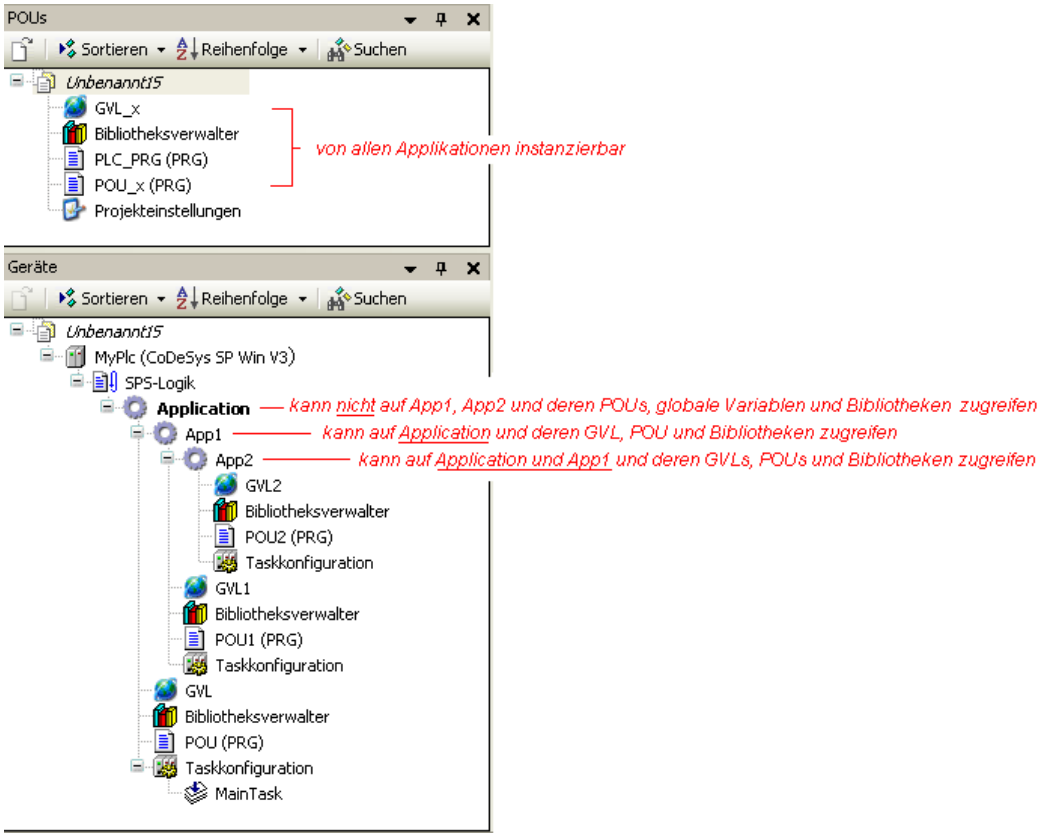


Abb. 21 "Vater"- und "Kindapplikationen"

- **Netzwerk-Scan der aktuellen Hardware**
  Als Erleichterung für das Anlegen der Steuerungskonfiguration im Gerätebaum unterstützen die Geräte-Editoren im Standardfall die Möglichkeit, die aktuelle Hardware abzutasten und in einem Dialog darzustellen. Der Anwender kann die gewünschten Module von dort direkt in den Gerätebaum übernehmen. Siehe dazu Befehl 'Geräte scannen'.



3.9 Applikation

- Eine "Applikation" umfaßt mehrere Objekte, die für das Ausführen einer bestimmten **Instanz des Steuerungsprogramms** auf einem bestimmten Gerät (Steuerung, SPS) nötig sind. Dazu können u.a. "unabhängige" Objekte, die im POUs-Fenster verwaltet werden, instanziert und einem Gerät im Gerätebaum zugewiesen werden. Dies entspricht der Denkweise der objektorientierten Programmierung. Aber auch rein applikationsspezifische POUs können von der Applikation verwendet werden.
- Eine Applikation wird durch ein **"Applikation"-Objekt** im Gerätebaum repräsentiert, das unterhalb eines SPS-Logik Knotens (d.h. programmierbares Gerät) eingefügt werden kann. Unterhalb einer Applikation können die Objekte, die das "Ressourcen-Set" der Applikation ausmachen, eingefügt werden.
- Alle Applikationen unterhalb desselben Geräts müssen **eindeutig benannt** sein.
- **Vor dem Einfügen** eines Applikationsobjekts über den Befehl Objekt hinzufügen muss das entsprechende programmierbare Gerät im Gerätebaum ausgewählt werden, ansonsten steht der Objekttyp 'Applikation' im 'Objekt hinzufügen'-Dialog nicht zur Auswahl. Vergeben Sie in diesem Dialog dann einen Namen für die neue Applikation und bestätigen Sie mit Open. Das Applikationsobjekt wird daraufhin im Gerätefenster unterhalb des Geräte-Objekts eingefügt.
- Ein essentielles Objekt für jede Applikation ist die **Taskkonfiguration** zur Kontrolle der Ausführung eines Programms (POU-Instanzen oder applikationsspezifische POUs). Zusätzlich können der Applikation Ressourcen-Objekte wie Globale Variablenlisten, Bibliotheken etc. direkt im Geräte-Fenster zugewiesen sein, die dann im Gegensatz zu den im POUs-Fenster verwalteten nur von dieser Applikation und ihren "Kindern" verwendet werden können.
- Innerhalb eines Projekts können **mehrere Applikationen für dasselbe** Gerät **definiert sein**, d.h. sie können auf der selben Steuerung laufen ("multi-application"). Dabei ist eine hierarchische Anordnung möglich ("Vater"- und "Kind"-Applikationen). In diesem Fall kann ein "Kind" die dem "Vater" zugewiesenen Objekte verwenden, nicht aber umgekehrt.

- Wenn mehrere Applikationen direkt unter einem Gerät hängen, kann für das **I/O-Mapping** dieses Geräts festgelegt werden, welche der Applikationen berücksichtigt werden soll.

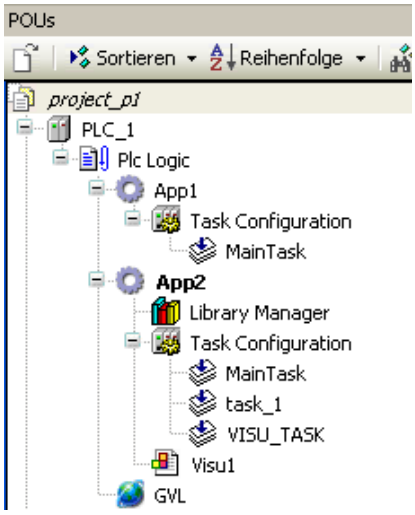


Abb. 22 Beispiel: Zwei Applikationen für ein Gerät

- Beachten Sie, dass die Applikation, mit der Sie im Online-Betrieb arbeiten wollen, als "**aktive Applikation**" gesetzt sein muss; siehe "App2" in der oben gezeigten Abbildung.
- Einer Applikation können im (Objekt-)Eigenschaften-Dialog bestimmte Eigenschaften gegeben werden; beispielsweise kann dynamische Speicherallozierung aktiviert werden.
- Beim Einloggen mit der Applikation auf dem Zielgerät (Steuerung oder Simulationsgerät) wird geprüft, welche Applikationen gerade auf der Steuerung vorliegen und ob die Applikations-Parameter auf der Steuerung mit denen des aktuellen Projekts übereinstimmen. Entsprechende Meldungen werden ausgegeben bzw. man erhält die Möglichkeit, **Applikationen auf der Steuerung zu löschen**.
- Beachten Sie den **Unterdialog 'Applikation'**des Geräte-Editors, in dem Sie sehen, welche Applikationen gerade auf einem Gerät vorhanden sind, und in dem man diese vom Zielsystem löschen kann. Hier können auch Applikationen angezeigt werden, die nicht durch ein separates Objekt im Gerätebaum vertreten sind, wie z.B. die "<Applikation>__symbols.app", die eine ggfs. für die Applikation <Applikation> erzeugte Symbolliste enthält.

3.10 Taskkonfiguration

Die Taskkonfiguration definiert eine oder mehrere Tasks zur Kontrolle und Ausführung des Applikationsprogramms auf der Steuerung.

Sie ist ein essentielles Ressourcen-Objekt für eine Applikation und muss im Gerätebaum unterhalb einer solchen eingefügt sein. Eine Task kann sowohl ein applikationszugeordnetes Programm aufrufen (POU, die im Projekt nur im Gerätebaum unterhalb der Applikation vorliegt), oder ein Programm, das im POUs-Fenster verwaltet wird. Im letzteren Fall ist dies eine "Instanzierung" einer projektweit verfügbaren POU.

Eine Taskkonfiguration kann im Task Editor, bearbeitet werden, wobei die verfügbaren Optionen vom Zielsystem abhängen.

Im Online-Betrieb bietet der Task Editor eine Monitoring-Ansicht mit Informationen über Zyklen, Zykluszeiten und Task-Status.

Als zusätzliche Funktionalität der Taskkonfiguration, wenn es vom Gerät unterstützt wird, ermöglicht die "Überwachung (Profiling)" eine dynamische Analyse der Bausteine, die von einer Task kontrolliert werden. Es gibt Auskunft über die Ausführungszeiten, die Anzahl der Bausteinaufrufe und die nicht durchlaufenen Code-Zeilen.

Wichtige Hinweise für Multitasking-Systeme:

Auf manchen System ist echtes preemptives Multitasking realisiert (z. B. PLC Designer SP). In diesem Falle ist folgendes zu beachten:

Wie in PLC Designer V2.3 teilen sich alle Tasks ein Prozessabbild. Grund: Ein eigenes Prozessabbild pro Task würde zu Lasten der Performance gehen. Das Prozessabbild kann jedoch somit immer nur zu einer Task konsistent sein. Beim Erstellen eines Projekts muss man deshalb als Anwender selbst dafür sorgen, dass bei Konflikten die Input-Daten in einen sicheren Bereich kopiert werden, bei Ausgängen gilt dasselbe. Möglichkeiten, die Konsistenz- und Synchronisierungsprobleme zu lösen, bieten beispielsweise die Bausteine der SysSem-Bibliothek.

Auch beim Zugriff auf andere globale Objekte (globale Variable, Bausteine) können in Multitasking-Systemen Konsistenzprobleme auftreten, wenn die Objekte die Datenbreite des Prozessors überschreiten (Strukturen oder Arrays, die eine logische Einheit bilden). Zur Lösung stehen hier wiederum z. B. die Bausteine der SysSem-Bibliothek zur Verfügung.

3.11 Kommunikation

3.11.1 Steuerungskonfiguration

Die "Steuerungskonfiguration" bildet die Ziel-Hardware im Programmiersystem ab, um die Ein- und Ausgänge und die Parameter der Steuerung und der Feldbusgeräte der Applikation zugänglich zu machen. Außerdem ermöglicht sie das Anzeigen der verfügbaren Geräte-Parameter.

Der Baum der "**Steuerungskonfiguration**", der im PLC Designer V2.3 in einem eigenen Editor gehandhabt wird, ist in V3.x **im Gerätebaum integriert**, in dem auch die anderen Objekte arrangiert sind, die nötig sind, um eine Applikation auf einem Zielsystem laufen zu lassen. Das Abbilden der aktuellen Hardware-Konfiguration im Gerätebaum wird in den Standard-Geräte-Editoren durch eine Scan-Funktionalität erleichtert.

Die Zuweisung der Ein- und Ausgänge der Steuerung zu Projektvariablen geschieht entweder mit Hilfe der "AT Deklaration" im Deklarationseditor oder im I/O-Abbild Dialog des Geräte-Editors, der die Dialoge zur Konfiguration eines Geräts bereitstellt.

Wenn unterhalb eines auf oberer Ebene des Gerätebaums positionierten Geräts mehrere Applikationen vorliegen, kann im I/O-Abbild (Mappping) Dialog dieses Geräts festgelegt werden, welche Applikation für das Mapping herangezogen werden soll. Wenn eine Steuerung neu in den Gerätebaum eingefügt wird, wird standardmäßig gleich die automatisch mit eingefügte Applikation als "Mapping-Applikation" eingetragen.

3.11.2 Netzwerk Topologie

Das Ziel ist ein System zu schaffen, das eine weitreichende Selbstkonfiguration (Adresszuweisung) zulässt, jedes Kommunikationsmedium transparent unterstützt und Datenpakete zwischen verschiedenen Netzwerken befördert. Der Routing Mechanismus sollte dabei so einfach sein, dass jeder Netzwerkknoten, also auch Knoten mit geringem Speicher, in der Lage ist Datenpakete umzuleiten. Daher sind umfangreiche Routing Tabellen, komplexe Rechnungen oder Anfragen während der Laufzeit zu vermeiden.

Das Steuerungsnetzwerk sollte hierarchisch aufgebaut sein, d.h. jeder Knoten darf einen Vaterknoten und eine beliebige Anzahl an Kindknoten besitzen. Ein Knoten ohne Vater ist ein sog. Top-level-Knoten. Zyklen sind nicht erlaubt, d.h. jedes Steuerungsnetzwerk hat die Struktur eines Baums.

Vater-Kind-Beziehungen entstehen durch die Spezifikation bestimmter Netzwerkbereiche. Ein Netzwerkbereich kann beispielsweise ein lokales Ethernet oder eine serielle Punkt-zu-Punkt-Verbindung sein. Wir unterscheiden das Hauptnetzwerk (mainnet) von den Subnetzwerken (subnet). Jeder Knoten gehört höchstens einem Hauptnetzwerk an, zu welchem auch sein Vaterknoten gehört, sofern ein solcher existiert. Für jeden Knoten kann eine beliebige Anzahl an Subnetzen konfiguriert werden, für welche der Knoten jeweils als Vater fungiert.

Würde ein Netzwerkbereich gleichzeitig als Subnetz von mehrerer Knoten definiert, so hätte er mehrere Väter. Die resultierende Konfiguration wäre damit jedoch ungültig, da ein Netzwerkbereich nur einen einzigen Vaterknoten haben darf.

3.11.3 Adressierung und Routing

Die Adressierung bildet die Topologie des Steurungsnetzwerks auf eindeutige Adressen ab. Eine **Knotenadresse** ist hierarchisch zusammengesetzt:

Für jede Netzwerkverbindung bestimmt der zugehörige Blocktreiber eine **lokale Adresse**, welche den Knoten innerhalb des lokalen Netzwerks eindeutig identifiziert. Für die gesamte Knotenadresse wird der lokalen Adresse der vom Vater vergebene **Subnetz Index** des lokalen Netzwerks vorangestellt. Diesem wird sodann die Knotenadresse des Vaters vorangestellt.

Dabei wird die Länge des Subnetzindices (in Bit) durch das Gerät bestimmt, wohingegen die Länge der lokalen Adresse durch den Netzwerktyp festgelegt ist.

Ein Knoten ohne Hauptnetzwerk ist ein Top-level-Knoten mit Adresse 0. Ein Knoten mit Hauptnetzwerk, welches keinen Vater enthält, ist ebenfalls ein Top-level-Knoten; er erhält die lokale Adresse des Hauptnetzwerks.

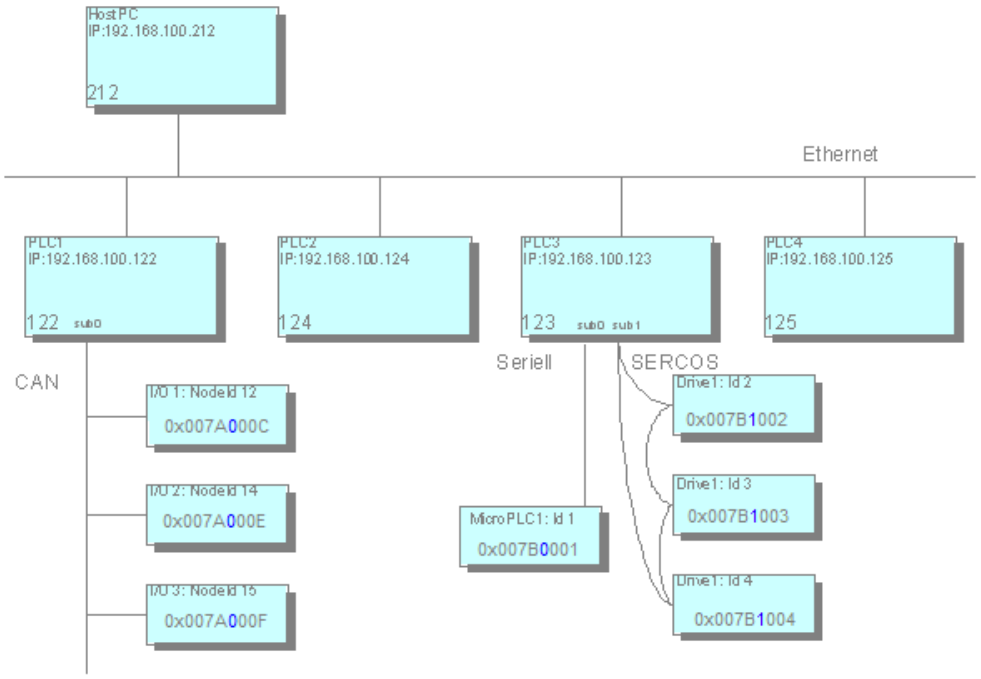


Abb. 23 Beispiel - Hauptnetzwerk und Subnetzwerke

Im Beispiel sind die Adressen der Knoten in hexadezimaler Schreibweise dargestellt. Die ersten 4 Ziffern repräsentieren die Adresse des jeweiligen Vaters im Hauptnetz, z.B. 0x007A=122 für PLC1. Das nächste Byte (in blauer Schrift) ist für den Subnetzindex reserviert und auf diesen folgt die lokale Adresse, z.B. C=12 für KnotenId 12.

Dank der Strukturierung der Adressen kann der Routing Algorithmus relativ schlank gehalten werden. So sind beispielsweise Routing Tabellen unnötig. Information wird nur lokal abgefragt: über die eigene Adresse und über die Adresse des Vaterknoten. Darauf basierend kann ein Knoten die Datenpakete korrekt bearbeiten:

- Falls die Zieladresse mit der Adresse des aktuellen Knoten übereinstimmt, so ist dieser als Empfänger bestimmt.
- Falls die Zieladresse mit der Adresse des aktuellen Knotens beginnt, so ist das Datenpaket entweder direkt für ein Kind oder für einen Nachkömmling des Knoten bestimmt und muss weitergeleitet werden.
- Andernfalls ist der Empfänger kein Nachkömmling des aktuellen Knoten und das Datenpaket muss an seinen eigenen Vater weitergeleitet werden.

Relative Adressierung ist eine Besonderheit: Relative Adressen enthalten nicht die Knotennummer des Empfängers, sondern beschreiben direkt den Pfad vom Sender zum Empfänger. Das Prinzip ist ähnlich zum relativen Pfad im Dateisystem: Die Adresse besteht aus der Anzahl der Schritte, die das Datenpaket nach oben transportiert werden muss, d. h. zum entsprechenden Vater, und aus dem anschließenden Pfad hinab zum Zielknoten.

Der Vorteil der relativen Adressierung liegt darin, dass zwei Knoten desselben Teilbaums die Kommunikation fortsetzen können, wenn der gesamte Teilbaum an eine andere Stelle im Gesamtnetzwerk verschoben wird. Während die absolute Knotenadressierung aufgrund dieser Verschiebung geändert werden muss, ist die relative Adressierung noch immer gültig.

Adressbestimmung

Die Kenntnis der eigenen Adresse erfordert von einem Knoten entweder die Kenntnis über die Adresse des Vaterknotens oder über die Tatsache ein Top-level-Knoten zu sein. Zu diesem Zweck versendet der Knoten während des Hochfahrens an alle Netzwerkteilnehmer eine Nachricht zur Adressbestimmung. So lange er keine Antwort auf diese Nachricht erhält, sieht sich der Knoten als Top-level-Knoten, fährt jedoch fort mit dem Versuch einen möglichen Vater zu entdecken. Ein Vaterknoten antwortet durch Bekanntgabe seiner Adresse. Damit wird der Knoten selbst seine Adresse vervollständigen und sie den Subnetzen bekanntgeben.

Eine Adressbestimmung kann während des Hochfahrens durchgeführt werden oder aber auf Anfrage des zur Programmierung benutzen PCs.

3.11.4 Adressstrukturen

Nachfolgend finden Sie eine detaillierte Beschreibung zum Aufbau der folgenden Adresstypen:

- Netzwerkadressen
- Knotenadressen
- Absolute und relative Adressen
- Broadcast Adressen

Netzwerkadressen

Netzwerkadressen repräsentieren eine Abbildung von Adressen vom Netzwerktyp (z.B. IP-Adressen) auf logische Adressen innerhalb eines Steuerungsnetzwerks. Diese Abbildung wird vom entsprechenden Blocktreiber vorgenommen. Innerhalb eines EtherNet mit Class C IP-Adressen sind die ersten 3 Bytes der IP-Adresse für alle Netzwerkgeräte gleich. Folglich genügen die letzten 8 Bit der IP-Adresse als Netzwerkadresse, zumal sie eine eindeutige Abbildung zwischen der beiden Adressen a,m Blocktreiber erlauben.

Ein Knoten hat für jede Netzwerkverbindung eine unterschiedliche Netzwerkadresse. Unterschiedliche Netzwerkverbindungen können dieselbe Netzwerkadresse haben, da jede Adresse nur lokal eindeutig sein muss.

Terminologie: Als Netzwerkadresse eines Knotens ohne Angabe der Netzwerkverbindung wird in der Regel die Netzwerkadresse im Hauptnetzwerk bezeichnet.

Die Länge einer Netzwerkadresse wird in Bit angegeben und kann vom Blocktreiber je nach Bedarf gewählt werden. Innerhalb eines Netzwerkbereichs muss für alle Knoten dieselbe Länge verwendet werden. Eine Netzwerkadresse wird von einem Array von Bytes dargestellt gemäß der nachfolgenden Codierung:

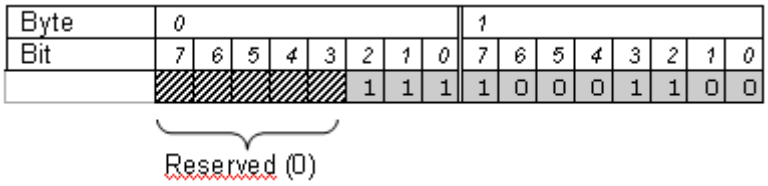
- Länge der Netzwerkadresse: n Bits
- Erforderliche Bytes: b = (n + 7) DIV 8
- Die (n MOD 8) Bits von niederstem Rang des ersten Bytes und alle übrigen (n DIV 8) werden für die Netzwerkadresse benutzt.

Beispiel - Netzwerkadresse

Länge: 11 Bit

Adresse 111 1000 1100

"Reserved" = reserviert



Knotenadressen

Die Knotenadresse gibt die absolute Adresse eines Knotens innerhalb des Steuerungsnetzwerkes an und ist daher eindeutig innerhalb des gesamten "Baums". Die Adresse setzt sich aus bis zu 15 Adresskomponenten zusammen, von denen jede jeweils 2 Bytes belegt. Je niedriger ein Knoten in der Netzwerkhierarchie angesiedelt ist, desto länger ist seine Adresse.

Die gesamte Knotenadresse besteht aus den Teiladressen aller vorangehenden Knoten und der Teiladresse des Knotens selbst. Jede Teiladresse besteht aus einer oder mehreren Adresskomponenten. Die Länge ist daher immer ein Vielfaches von 2 Bytes. Die Teladresse eines Knotens besteht aus der Netzwerkadresse des Knotens in seinem Hauptnetzwerk und dem Subnetzindex des Hauptnetzwerkes beim Vater. Die für den Subindex erforderlichen Bits werden vom Router des Vaterknotens festgelegt. Zwischen dem Subnetzindex und der Netzwerkadresse können Füllbits eingefügt werden um zu erreichen, dass die Länge der Teiladresse ein Vielfaches von 2 Bytes ist.

Spezialfälle:

- Ein Knoten ohne Hauptnetzwerk: Das bedeutet, dass es weder einen Subnetzindex noch eine Netzwerkadresse im Hauptnetzwerk gibt. In diesem Fall wird die Adresse auf 0x0000 gesetzt.
- Ein Knoten im Hauptnetzwerk ohne Vater: In diesem Fall wird ein Subnetzindex der Länge 0 angenommen. Die Teiladresse entspricht der Netzwerkadresse, bei Bedarf erweitert um Füllbits.

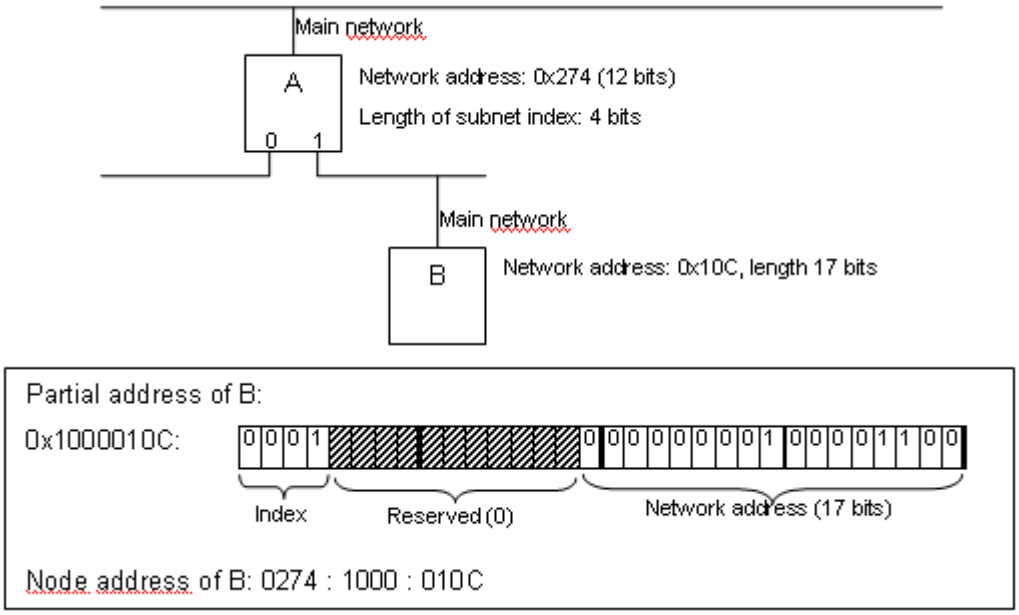


Abb. 24 Beispiel - Knotenadressen

Die Knotenadresse wird immer hexadezimal angegeben. Die einzelnen Adresskomponenten (jeweils von 2 Bytes) werden durch einen Doppelpunkt ":" getrennt. Da dies ein Array von Bytes darstellt und keinen 16-bit Wert, werden die Komponenten nicht im Intel-Format angezeigt. Für händisch eingegebene Adressen werden fehlende Stellen in einer Adresskomponente durch führende Nullen ergänzt: "274" = "0274". Um die Lesbarkeit zu verbessern sollte die Ausgabe auch immer die führenden Nullen enthalten.

Absolute und relative Adressen

Die Kommunikation zwischen zwei Knoten kann auf relativen oder absoluten Adressen basieren. Absolute Adressen sind identisch mit Knotenadressen. Relative Adressen geben den Pfad vom Sender zum Empfänger an. Sie bestehen aus einem Adressen-Offset und dem absteigenden Pfad hin zum Empfänger.

Der (negative) Adressen-Offset beschreibt die Anzahl an Andresskomponenten um die ein Datenpaket im Baum nach oben durchgereicht werden muss, bevor es vom gemeinsamen Vater wieder nach unten gereicht werden kann. Da Knoten Teiladressen benutzen können, die aus mehr als einer Komponente bestehen, ist die Anzahl der zu passierenden Vaterkomponenten immer gleich dem Adress-Offset. Das bedeutet, dass die Abgrenzung zwischen den Vaterknoten nicht länger eindeutig ist, weswegen der gemeinsame Anfang der Adressen der Kommunikationspartner als Vateradresse verwendet wird. Jede Adresskomponente wird als Aufwärtsschritt gezählt, unabhängig vom aktuellen Vaterknoten. Jeder der durch diese Annahme entstehenden Fehler kann vom entsprechenden Vaterknoten entdeckt werden und muss von ihm entsprechend behandelt werden.

Nach Erreichen des gemeinsamen Vaters wird wie gewohnt der relative Pfad (als Array von Adresskomponenten) nach unten verfolgt.

Formal: Die Knotenadresse des Empfängers wird gebildet durch Entfernen der letzten Adressen-Offset Komponenten aus der Knotenadresse des Senders und durch Anhängen des relativen Pfads an die verbleibende Adresse.

Beispiel:

Im Beispiel ist jede Adresskomponente durch einen Buchstaben repräsentiert, ein Punkt trennt jeweils Knoten voneinander. Da ein Knoten mehrere Adresskomponenten tragen kann, kann er im Beispiel mehrere Buchstaben haben.

Knoten A: a.bc.d.ef.g

Knoten B: a.bc.i.j.kl.m

- Adresse des niedrigsten gemeinsamen Vaters: a.bc
- Relative Adresse von A nach B: -4/i.j.kl.m (Die Zahl 4 resultiert aus den 4 Komponenten d, e, f, und g, die das Datenpaket nach oben durchgereicht werden muss)

Um ein korrektes Arbeiten des Routings zu garantieren muss die relative Adresse bei jedem Passieren eine Zwischenknotens angepasst werden. Es ist ausreichend den Adress-Offset anzupassen. Dies wird immer vom Vaterknoten durchgeführt: Wenn ein Knoten ein Datenpaket von einem seiner Subnetze erhält, so wird der Adress-Offset um die Länge der Adresskomponenten dieses Subnetztes erhöht:

- Ist der neue Adress-Offset < 0, so muss das Datenpaket weiter nach oben gereicht werden.
- Ist der Adress-Offset >= 0, so muss das Datenpaket an den Kindknoten gereicht werden, dessen lokale Adresse vom Adress-Offset ausgehend dem relativen Pfad entspricht. Zunächst muss jedoch das Adress-Offset um die Länge der lokalen Adresse des Kindes erhöht werden, damit der Kindknoten die korrekte Adresse sieht.

Eine spezielle Situation entsteht, wenn der oben beschriebene Fehler beim Bestimmen des gemeinsamen Vaters eintritt. In diesem Fall ist der Adrss-Offset des wirklichen Vaterknotens negativ, aber dieser Wert ist größer als die Länger der Teiladressen des Subnetzes, von denen das Paket stammt. Der Knoten muss diesen Fehler aufdecken, auf Basis der Adresse des Vorgängerknoten und der Längendifferenz die lokale Adresse des Kindknoten berechnen und den Adress-Offset entsprechend anpassen, so dass der nächste Knoten eine korrekte relative Adresse sieht. Auch in diesem Fall bleiben die Adresskomponenten an sich unverändert, nur der Offset wird geändert.

Broadcast Adressen

Es gibt zwei Arten von Broadcast - globale und lokale. Ein globaler Broadcast wird an alle Knoten eines Netzwerks gesandt. Für diesen Zweck ist die leere Knotenadresse (Länge 0) reserviert.

Lokale Broadcasts werden an alle Geräte eines Netzwerkbereichs geschickt. Zu diesem Zweck werden alle Bits der Netzwerkadresse auf 1 gesetzt. Das ist sowohl in relativen wie auch in absoluten Adresse zulässig.

Ein Blocktreiber muss beide Arten von Broadcastadressen bearbeiten können, d. h. leere Netzwerkadressen ebenso wie die Netzwerkadressen, deren Bits sämtlich auf 1 gesetzt wurden, müssen interpretiert und an alle betroffenen Teilnehmer versandt werden.

3.11.5 Data Server

Ein "Data Server" kann einer Applikation hinzugefügt werden, um entfernte Datenquellen nutzen zu können. "Entfernte Datenquelle" in diesem Zusammenhang bedeutet, dass Variablen ("data items"), die in anderen Geräten definiert und verwendet werden, auch in der lokalen Applikation genutzt werden können.

Im Gegensatz zum Datenaustausch über Netzwerkvariablen (broadcasting) baut der Data Server Punkt-zu-Punkt-Verbindungen auf. In Abhängigkeit vom Zugriffs-Flag der auszutauschenden Datenverbindungen werden diese in der Datenquelle bzw. in der aktuellen Applikation jedesmal aktualisiert, wenn sich ihr Wert auf der jeweils anderen Seite ändert.

Die Benutzung eines Data Servers ist die schnellere Alternative zur Bereitstellung von Daten mittels einer Symbolkonfiguration. Derzeit kann über den Data Server noch nicht auf Daten zugegriffen werden, die von einem OPC-Server bereitgestellt werden. In diesem Fall ist die Implementation einer Symbolkonfiguration noch die geeignete Vorgehensweise.

3.11.6 Netzwerkvariablen

Wenn es vom Zielsystem unterstützt wird, kann innerhalb eines Netzwerks Datenaustausch mittels Netzwerkvariablen betrieben werden.

Netzwerkvariablen müssen in starren Variablenlisten sowohl im Sender als auch Empfänger (in einem oder mehreren Projekten) definiert werden. Ihre Werte werden via **Broadcasting** im Netz versendet. Beachten Sie, dass sich dies grundsätzlich vom Datenaustausch mittels eines Data Servers unterscheidet, der definierte Punkt-zu-Punkt-Verbindungen zwischen lokaler Applikation und entfernten Datenquellen verwendet.

Netzwerkvariablen werden gehandhabt in....

- Globalen Variablenlisten (**GVL**) im sendenden Gerät (**Sender**) und in einer bzw. mehreren entsprechenden
- Globale Netzwerkvariablenliste (**GNVL**) in einem bzw. mehreren empfangenden Gerät(en) (**Empfänger**). GNVLs werden im Netzwerkvariablenlisten-Editor angezeigt.

Die zueinander gehörigen GVL und GNVLs müssen dieselben Variablendeklarationen enthalten.

Eine Globale Variablenliste (GVL), die Netzwerkvariablen definieren soll, muss spezielle Netzwerkeigenschaften haben. Dies sind Protokoll- und Übertragungsparameter, gemäß denen die Variablenwerte innerhalb des Netzwerks versendet werden und von allen Geräten mit einer entsprechenden GNVL empfangen werden können.

Beachten Sie, dass die Übertragung von Netzwerkvariablen immer **nur in eine Richtung** erfolgt, vom Sender (GVL) zum Empfänger (GNVL)! Allerdings kann dadurch dass jedes Gerät sowohl GVLs als auch GNVLs handhaben kann, erreicht werden, dass ein Gerät sowohl als Sender als auch als Empfänger agiert. Eine Sender GVL kann innerhalb desselben oder von einem anderen Projekt bereitgestellt werden. Beim Anlegen einer GNVL kann die Sender GVL entweder aus der Auswahlliste der verfügbaren GVLs innerhalb des Netzwerks genommen werden oder sie kann aus einer Exportdatei eingelesen werden, die vorher aus der GVL erzeugt wurde (beispielsweise über den Verknüpfung mit Datei Eigenschaften Dialog). Eine solche Exportdatei ist auf jeden Fal nötig, wenn die Sender GVL in einem anderen Projekt definiert ist.

Als Voraussetzung für den Austausch von Netzwerkvariablen müssen die entsprechenden **Netzwerkbibliotheken** installiert sein. Dies erfolgt für die Standard-Netzwerkfunktionalitäten, wie z.B. für UDP, automatisch sobald die Netzwerkeigenschaften einer GVL gesetzt werden.

Beispiel:

Im folgenden Beispiel wird der Aufbau eines einfachen Netzwerkvariablen-Austauschs beschrieben. Im Sender-Gerät wird eine GVL angelegt, im Empfänger die entsprechende GNVL:

Vorbereitende Arbeiten in einem Standard-Projekt, in dem ein Sendergerät Dev_Sender und ein Empfängergerät Dev_Empfänger im Gerätebaum angelegt sind:

- Legen Sie eine POU (Programm) "prog_sender" unterhalb "Applikation" in Dev_Sender an.
- In der Taskkonfiguration dieser Applikation fügen Sie die Task "Task_S" hinzu, die "prog_sender" aufruft.
- Legen Sie eine POU (Programm) "prog_receiver" unterhalb "Applikation" in Dev_Receiver an.
- In der Taskkonfiguration dieser Applikation fügen Sie die Task "Task_R" hinzu, die "prog_receiver" aufruft.

1. Definieren der Globalen Variablenliste im Sender-Gerät:

Selektieren Sie 'Applikation' unterhalb 'Dev_Sender'. Über den 'Objekt hinzufügen'-Dialog **fügen Sie eine Globale Variablenliste GVL_Sender** ein. Füllen Sie die Liste mit den folgenden Variablendeklarationen:

```
VAR_GLOBAL
iglobvar:INT;
bglobvar:BOOL;
strglobvar:STRING;
END_VAR
```

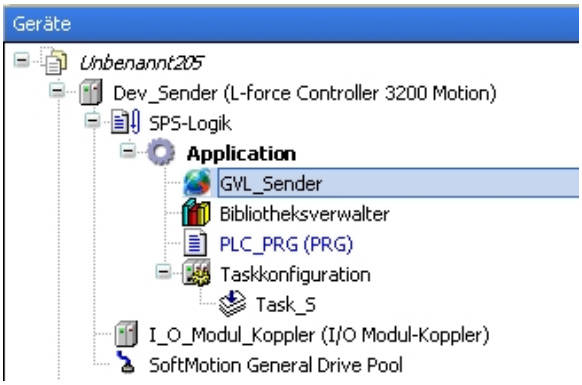


Abb. 25 Einfügen einer GVL im Sendergerät

2. Definieren Sie die Netzwerkeigenschaften der Sender GVL:

Wählen Sie 'GVL_Sender' im Gerätebaum aus und wählen im Kontextmenü den Befehl **Eigenschaften**. Stellen Sie die Netzwerkeigenschaften folgendermaßen ein:

Abb. 26 Setzen der GVL Netzwerkeigenschaften

3. Anlegen einer Globalen Netzwerkvariablenliste im Empfänger:

Wählen Sie 'Applikation' unter Gerät 'Dev_Receiver'. Öffnen Sie den Dialog 'Objekt hinzufügen' um eine **globale Netwerkvariablenliste hinzuzufügen** die das Gegenstück zur oben definierten Sender GVL bildet:

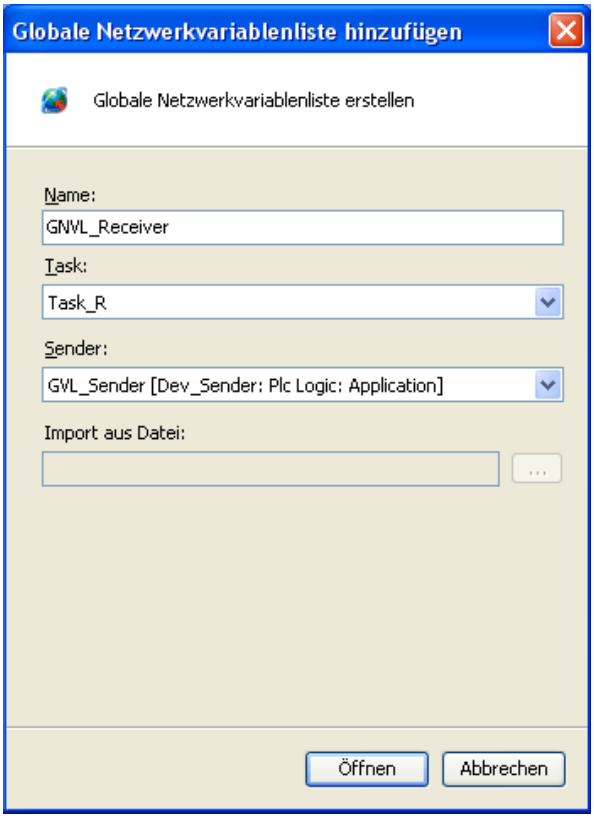


Abb. 27 Anlegen einer GNVL im Empfängergerät

Geben Sie bei Name "GNVL_Receiver" ein. Im Feld 'Sender' gibt es eine Auswahlliste aller gerade im Projekt verfügbaren GVLs mit Netzwerkeigenschaften Für das vorliegende Beispiel findet sich hier nur "GVL_Sender". (Nur zur Information: Wenn Sie eine GVL aus einem anderen Projekt verwenden wollten, müßte eine aus dieser Liste erzeugte Exportdatei vorliegen, die dann hier mit "Import aus Datei" importiert werden könnte)

Schließen Sie den Dialog mit OK, um das Objekt 'GNVL_Receiver' unterhalb 'Applikation' in 'Device_Receiver' einzufügen. Diese GNVL enthält automatisch dieselben Variablendeklarationen wie 'GVL_Sender':

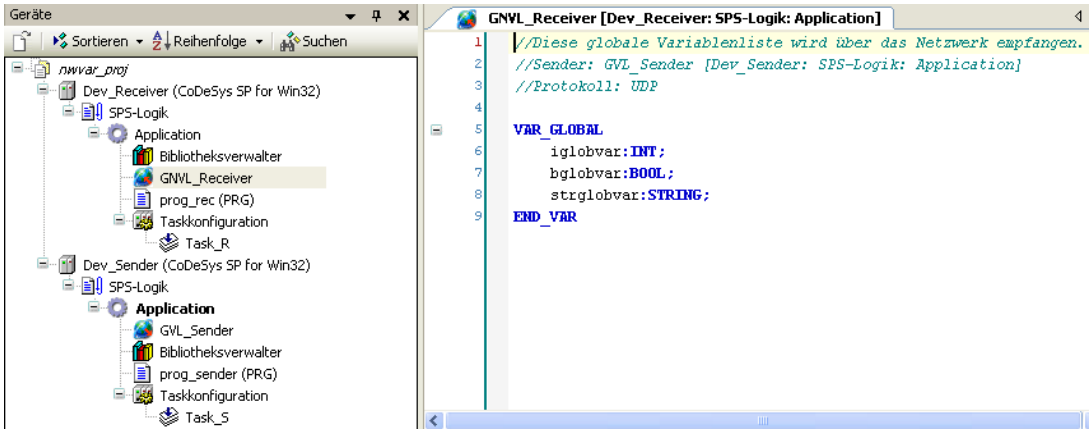


Abb. 28 GNVL_receiver enthält die Variablendeklarationen von GVL_Sender

4. Optional: Prüfen bzw. verändern Sie die Netzwerkeinstellungen der Globalen Netzwerkvariablenliste:

Wenn erforderlich, können die **Netzwerkeinstellungen** der GNVL, die beim Anlegen dieser Liste definiert worden waren, erneut angezeigt und verändert werden. Dazu wählen Sie 'GNVL_Receiver' im Gerätebaum aus und öffnen über das Kontextmenü den Eigenschaften-Dialog und dort die 'Netzwerkeinstellungen'.

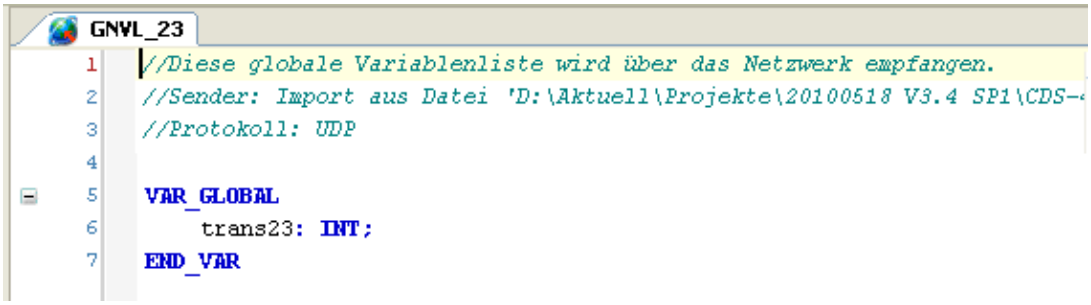
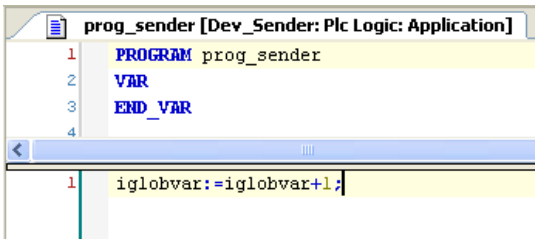


Abb. 29 GMVL Netzwerkeinstellungen

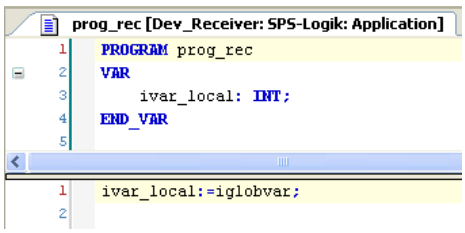
5. Testen Sie den Netzwerkvariablenaustausch:

Um die Netzwerkvariablen online zu testen, führen Sie die folgenden Schritte durch:

- In prog_sender in der Sender-Applikation geben Sie folgende Verwendung von Variable iglobvar ein:



- In prog_rec in der Empfänger-Applikation geben Sie folgende Verwendung der Netzwerkvariablen iglobvar ein:



- Verbinden Sie Sender- und Empfänger-Applikationen mit dem Netzwerk und starten Sie die Applikationen. Beobachten Sie in den Online-Ansichten der Bausteine, ob die Werte von iglobvar im Empfänger mit denen im Sender übereinstimmen.

Netzwerkvariablen-Kommunikation zwischen V2.3 und V3 Steuerungen:

Eine Kommunikation mit Hilfe von Netzwerkvariablen ist auch möglich, wenn die teilnehmenden Steuerungen mit Applikationen aus verschiedenen Versionen des Programmiersystems arbeiten (V2.3 <-> V3). In diesem Fall kann jedoch nicht der Export/Import-Mechanismus für das Anlegen der genau übereinstimmenden Variablenlisten verwendet werden, die im Sender- und Empfängerprojekt benötigt werden. Dies kommt daher, dass die V2.3 und V3 Variablenexportdateien (*.exp vs *.gvl) unterschiedliche Informationen enthalten. Wenn eine lesende GNVL in V3 aufgesetzt wird, muß die entsprechende Netzwerkparameterkonfiguration in Form einer *.gvl-Datei vorliegen, die zuvor aus dem V3 Sender exportiert wurde. Eine aus einem V2.3 Sender exportierte *.exp-Datei enthält diese Information aber nicht.

Mögliche Lösung für einen Netzwerkvariablenaustausch zwischen V2.3 und V3 Applikationen:

1. Bilden Sie die V2.3 NVL in V3 nach (Hinzufügen einer GVL mit Netzwerkeigenschaften, die die gleichen Variablendeklarationen enthält wie die 2.3 NVL)

2. Exportieren Sie die neue GVL in eine *.exp-Datei ("Verknüpfung mit Datei" Eigenschaften)

Hinweis: Aktivieren Sie die Option "Vom Übersetzen ausschließen" und Sie können die GVL im Projekt behalten ohne Precompile-Fehler und uneindeutige Namen zu erhalten. Deaktivieren Sie die Option, wenn nach erforderlichen Änderungen in der GVL die exp-Datei neu erzeugt werden muss.

3. Reimportieren Sie die Liste, d.h. legen Sie eine neue GNVL unter Verwendung der vorher erzeugten *.exp-Datei an, um eine korrekt passende Variablenliste im Empfänger zu erhalten.

Beispiel

Es gibt ein Projekt 23.pro mit einer globalen Variablenliste GVL_23, die die folgenden Deklarationen enthält: VAR_GLOBAL trans23: INT; END_VAR.

Variable trans23 soll von einer V3-Applikation gelesen werden können.

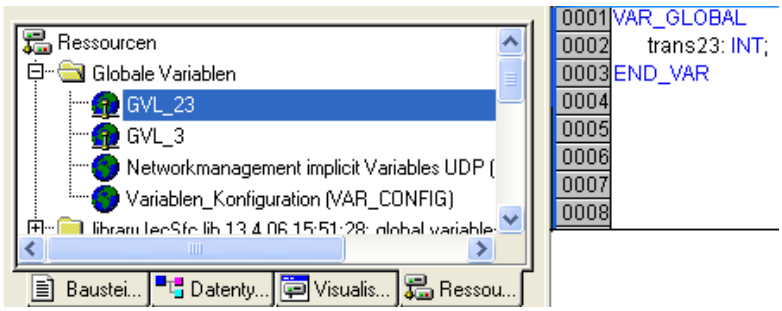


Abb. 30 GVL in V2.3 Projekt

Die Netzwerkeigenschaften von GVL_23 sind folgendermaßen konfiguriert:

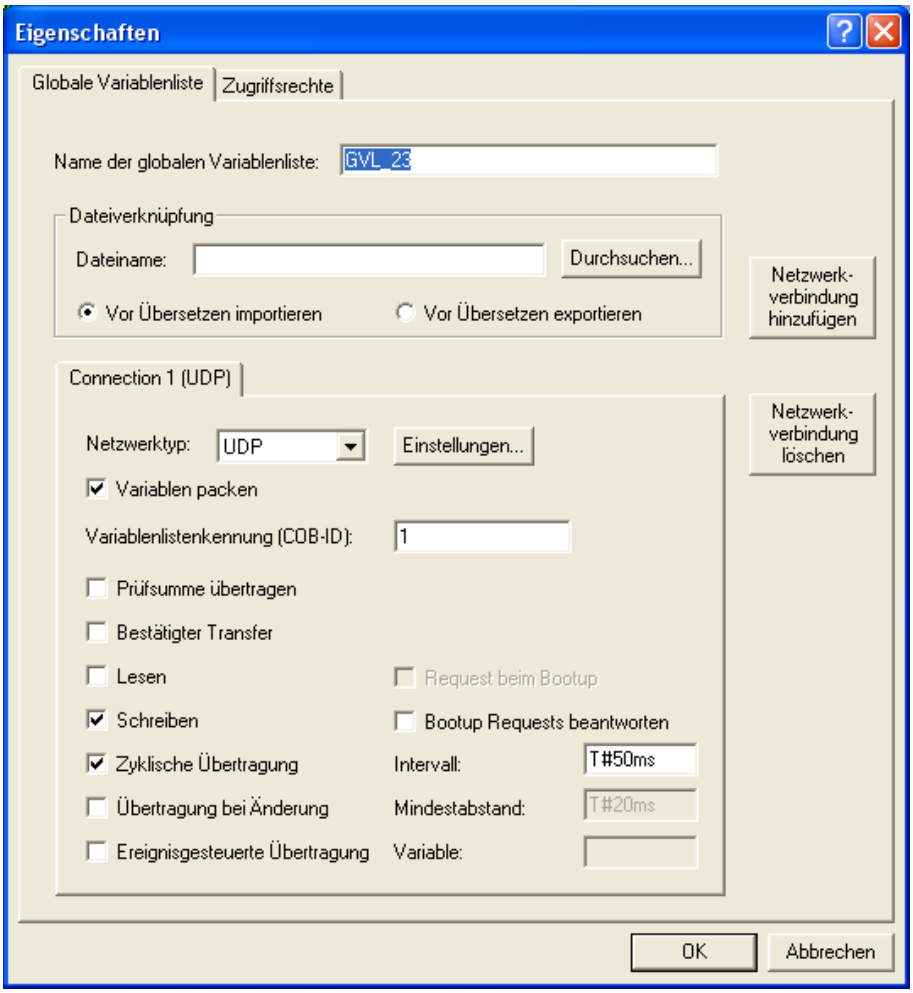


Abb. 31 Eigenschaften von GVL_23

Wenn Sie GVL_23 in eine *.exp-Datei exportieren würden, dann würde diese Datei nur die Deklaration VAR_GLOBAL trans23: INT; END_VAR; enthalten. Deshalb müssen Sie zunächste GVL_23 in V3 nachbilden (siehe GVL_23 in der Abbildung oben):

Fügen Sie in einem V3 Projekt ein GVL Objekt mit Namen "GVL_23" unterhalb einer Applikation ein und führen Sie folgende Schritte aus:

- Setzen Sie die Netzwerkeigenschaften wie in 23.pro definiert
- Geben Sie in den "Mit Datei verknüpfen"-Eigenschaften eine Exportdatei "23.gvl" an
- Empfehlung: Setzen Sie die Option "Vom Übersetzen ausschließen" (Beschreibung siehe oben in 2.)
- Kompilieren Sie das V3 Porjekt, um die 23.gvl-Datei zu erzeugen (enthält dann Variablendefinitionen + Konfigurationsdaten !)

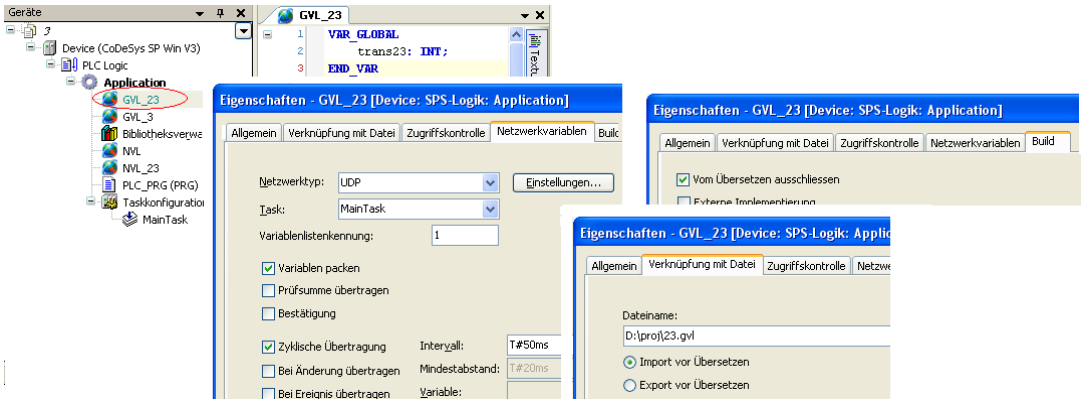


Abb. 32 Nachbilden der GVL in V3

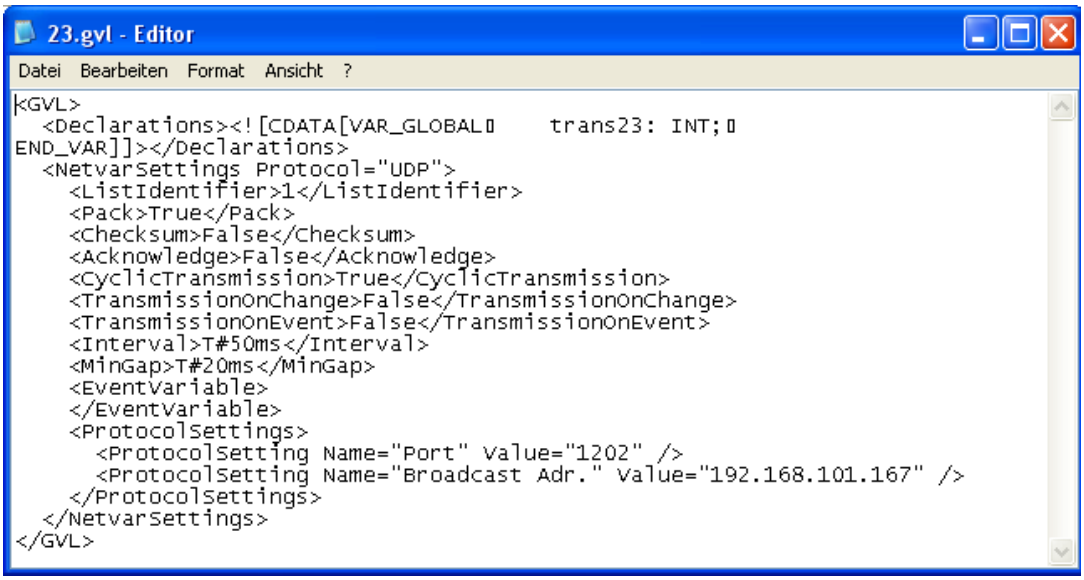


Abb. 33 Resultierende Exportdatei "23.gvl"

Fügen Sie nun mit Hilfe der 23.gvl-Datei im V3 Projekt ein GNVL Objekt ein (Option "Von Datei importieren"), welches ermöglicht die Variable trans23 aus der 2.3 Steuerung zu lesen.

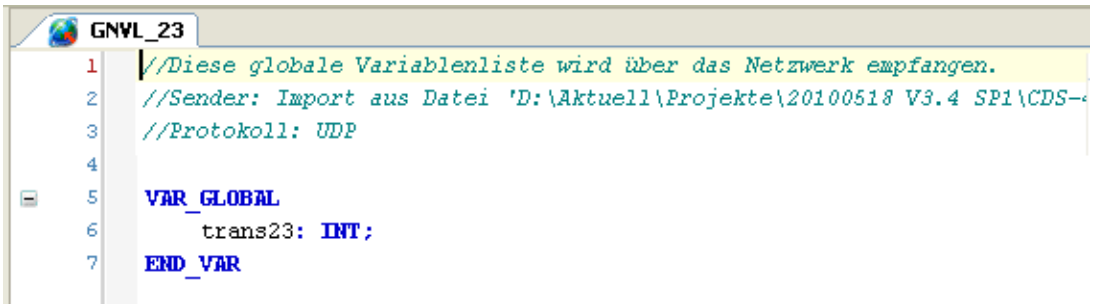


Abb. 34 GNVL in V3 Projekt

Wenn nun sowohl das V2.3 Projekt als auch die V3 Applikation innerhalb des Netzwerks laufen, kann die V3 Applikation Variable trans23 aus Projekt 23.pro lesen

3.12 Code-Generierung und Online Change

Code-Generierung, Übersetzungsprotokoll

Maschinen-Code wird erst generiert, wenn die Applikation auf das Zielgerät (Steuerung (SPS) oder Simulationsgerät) geladen wird. Bei jedem Download wird das Übersetzungsprotokoll, welches Code und Identifikation der geladenen Applikation enthält, als Datei "**<projectname>.<devicename>.<application ID>.compileinfo**" im selben Verzeichnis wie das Projekt gespeichert. Das Übersetzungsprotokoll wird gelöscht bei Ausführung des Kommandos <application> bereinigen resp. Alles bereinigenl.

Beachten Sie, dass kein Maschinen-Code erzeugt wird, wenn das Projekt mit den 'Erstellen'-Befehlen übersetzt wird. Dieser Übersetzungsprozeß dient dazu, das Projekt auf syntaktische Fehler zu überprüfen. Diese werden im Meldungsfenster (Meldungskategorie 'Übersetzen') ausgegeben.

Online Change

WARNUNG!
Online Change verändert das laufende Applikationsprogramm und bewirkt keinen Neustart.
Stellen Sie sicher, dass der neue Applikations-Code dennoch das gewünschte Verhalten des gesteuerten Systems bewirkt.
Abhängig von der gesteuerten Anlage können Schäden an Anlage und Werkstücken entstehen oder Gesundheit und Leben von Personen gefährdet sein.

Hinweis:
1. Wenn ein Online Change durchgeführt wird, werden die applikationsspezifischen Initialisierungen (Referenzfahrt etc.) nicht ausgeführt, weil die Maschine ihren Status beibehält. Aus diesem Grund könnte der neue Programm-Code nicht den gewünschten Effekt haben.
2. Zeigervariablen behalten ihren Wert aus dem letzten Zyklus. Wenn ein Zeiger auf eine Variable zeigt, die durch den Online Change ihre Größe verändert hat, wird der Wert nicht mehr korrekt geliefert. Stellen Sie sicher, dass Zeigervariablen in jedem Zyklus erneut zugewiesen werden.

Wenn die Applikation, die gerade auf der Steuerung läuft, seit dem letzten Download im Programmiersystem verändert wurde, werden beim Online Change nur die veränderten Objekte des Projekts auf die Steuerung geladen, während das Programm dort weiter läuft.

Es gibt zwei Wege einen Online Change durchzuführen:

1. Sobald Sie versuchen, mit einem veränderten Applikationsprogramm erneut einzuloggen, werden Sie gefragt, ob ein Online Change durchgeführt werden soll:

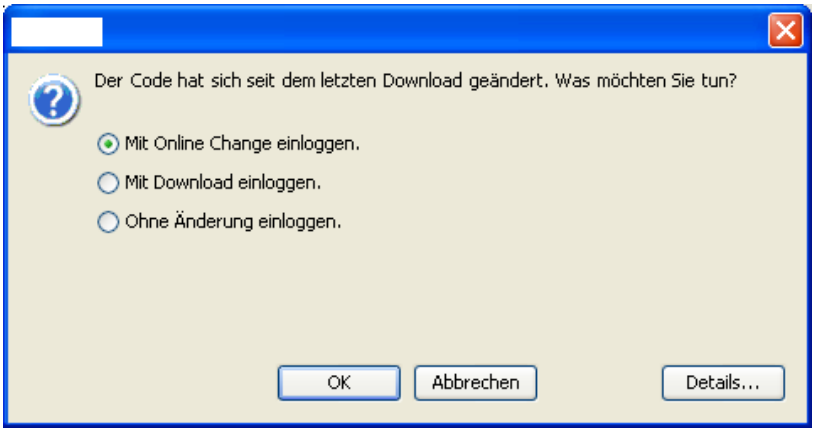


Abb. 35 Online Change Dialog

Mit Online Change einloggen: Diese Option ist standardmäßig aktiviert. Wenn also der Dialog mit **OK** verlassen wird, so werden die geänderten Objekte geladen und unmittelbar in der Online-Ansicht (Monitoring) des jeweiligen Objekts angezeigt.

Mit Download einloggen: Diese Option muss aktiviert sein, wenn das Projekt komplett übersetzt und geladen werden soll.

Ohne Änderung einloggen: Diese Option muss aktiviert sein, wenn das Programm auf der Steuerung unverändert weiterlaufen und die neuen Änderungen nicht geladen werden sollen. Danach kann explizit ein Download (<Applikation> laden) durchgeführt werden, der das gesamte Projekt neu lädt, oder beim nächsten Einloggen wird wieder gefragt, ob ein Online Change durchgeführt werden soll.

Die Schaltfläche Details im Login-Dialog führt zu Informationen (Projektname, Letzte Änderung, IDE Version, Autor, Beschreibung) über die aktuelle Applikation im IDE (integrated development environment = Programmiersystem) im Vergleich zu derjenigen, die gerade auf der Steuerung (SPS) liegt:

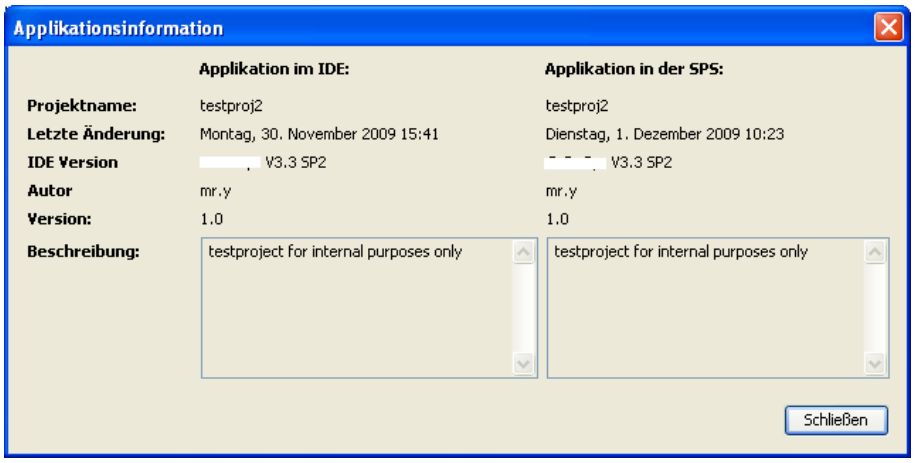


Abb. 36 Applikationsinformation, Dialog

Wenn gerade andere Applikationen auf der SPS liegen, erscheinen weitere Dialogboxen zur Handhabung der Situation. Eine Beschreibung hierzu findet sich in: Einloggen.

2. Mit dem Befehl Online Change <Applikation> können Sie einen Online Change auf einer laufenden Applikation explizit herbeiführen.

Beachten Sie, dass Online Change eines geänderten Projekts für eine Applikation nicht mehr möglich ist, nachdem ein '**Bereinigen**' durchgeführt wurde (Alles bereinigen, <Applikation> bereinigen). In diesem Fall wird die Information über die seit dem letzten Download geänderten Objekte gelöscht. Somit kann nur noch das gesamte Projekt neu geladen werden.

Beachten sie folgende Punkte, bevor Sie einen Online Change durchführen:

Ist der geänderte Code fehlerfrei?

Vorsicht: Applikationsspezifische Initialisierungen (Referenzfahrt etc.) werden nicht ausgeführt, da die Maschine ihren Status beibehält. Kann der neue Programm-Code wirklich ohne eine Re-Initialisierung arbeiten?

Vorsicht: Pointer-Variablen behalten Ihren Wert aus dem letzten Zyklus. Wenn auf eine Variable gezeigt wird, die nun ihre Größe geändert hat, wird der Wert nicht länger korrekt sein. Deshalb sollten Sie sicherstellen, dass Pointer-Variablen in jedem Zyklus erneut zugewiesen werden.

Vorsicht: Wenn der aktive Schritt in einem AS-Diagramm entfernt wird, bleibt das Diagramm inaktiv.

Bootapplikation (Bootprojekt)

Momentan: Bei jedem Download wird automatisch die aktive Applikation als Datei "application.app" im Zielsystemverzeichnis gespeichert. Der Befehl 'Bootapplikation erzeugen' erlaubt auch im Offline-Modus, die Bootapplikation in eine Datei zu speichern.

Eine Bootapplikation wird automatisch gestartet, wenn die Steuerung gestartet wird ("boot"). Zu diesem Zweck muss das Applikationsprojekt auf der Steuerung in einer Datei <Projektname>.app verfügbar sein. Um die Datei zu erzeugen, verwenden Sie den Befehl Bootapplikation erzeugen (Befehlskategorie 'Online').

3.13 Monitoring

Im Online-Modus gibt es verschieden Möglichkeiten, die aktuellen Werte der Variablen eines Objekts auf der Steuerung anzuzeigen:

- **Inline Monitoring** im Implementations-Editor eines Objekts. Details finden Sie in der Beschreibung des jeweiligen Editors..
- **Online Ansicht des Deklarationseditors** eines Objekts. Details finden Sie in der Beschreibung des Deklarationseditors.
- **Objektunabhängige Überwachungslisten**. Details finden sie in der Beschreibung der Überwachungs-Fenster.
- **Trace-Kurven**. Aufzeichnen und Anzeigen von Variablenwerten auf der Steuerung. Details finden Sie in der Beschreibung der Trace-Funktionalität.
- **Rezepturen**. Benutzerdefiniertes Variablen-Set zum Setzen und Überwachen dieser Variablen auf der Steuerung. Siehe Rezepturverwaltung.

3.14 Debugging / Fehlersuche

Zur Untersuchung von Programmierfehlern können die Debugging-Funktionen im PLC Designer verwendet werden. Beachten Sie in diesem Zusammenhang die Möglichkeit eine Applikation im **Simulationsmodus**, also ohne notwendige Anbindung an ein echtes Zielgerät.

Haltepunkte (Breakpoints) können an bestimmten Positionen im Programm gesetzt werden um einen Ausführungsstop zu erzwingen. Bestimmte Bedingungen, nämlich welche Task(s) betroffen sind, und in welchen Zyklusabständen der Haltepunkt wirksam werden soll, können für jeden Haltepunkt definiert werden.

Einzelschrittabarbeitung ermöglicht, das Programm in kontrollierten Schritten zu durchlaufen.

An jedem Stop, definiert durch Schrittmarken bzw. Haltepunkte können die jeweiligen Variablenwerte untersucht werden.

Haltepunkte

Ein **Haltepunkt (Breakpoint)**, der in einem Applikationsprogramm gesetzt ist, veranlaßt einen Stop bei der Ausführung des Programms. Die möglichen Haltepunktpositionen hängen vom jeweiligen Programm-Editor ab. Auf jeden Fall gibt es immer eine Haltepunktposition am Ende einer POU.

Ein wichtiges Werkzeug dabei ist der Haltepunkte-Dialog, in dem alle momentan definierten Haltepunkte aufgelistet werden und in dem Haltepunkte hinzugefügt, gelöscht und modifiziert werden können.

Neu: Bedingte Haltepunkte; der Stop am Haltepunkt kann davon abhängig gemacht werden, welche Task gerade ausgeführt wird oder der wievielte Zyklus gerade durchlaufen wird.

Symbole:

Haltepunkt aktiviert

Haltepunkt deaktiviert

Stop an Haltepunkt im Online-Modus

Einzelschrittabarbeitung (Stepping)

Einzelschrittabarbeitung ermöglicht eine kontrollierte Ausführung eines Applikationsprogramms, z.B. zum Zweck der Fehlersuche. Grundsätzlich wird durch wiederholtes Betätigen der Taste <F10> von Anweisung zu Anweisung gesprungen, es können jedoch auch aufgerufene Bausteine übersprungen werden oder....

... neu im Vergleich zu PLC Designer V2.3:

- Die Anweisung, die als nächste ausgeführt werden soll, kann explizit definiert werden ('Nächste Anweisung festlegen').
- Der nächste Ausführungsstop kann einfach dadurch festgelegt werden, dass der Mauszeiger an der gewünschten Stelle platziert wird ('Ausführen bis Cursor').
- 'Ausführen bis Rücksprung' bewirkt einen Schritt zurück zum letzten Aufrufer.

Symbole:

augenblickliche Position beim Steppen, angezeigt durch einen gelben Pfeil vor der Zeile und einer gelben Schattierung der zugehörigen Operation, Beispiel

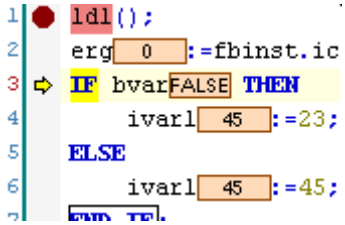


Abb. 37 Beispiel: vom Haltepunkt wurde mit Befehl 'Einzelschritt' zur nächsten

3.15 Drucken

Die Ansicht des gerade aktiven Editors kann mit Hilfe der Funktion Drucken ausgedruckt werden.

Beachten Sie alternativ die Möglichkeit, eine "Dokumentation" ausgewählter oder aller Objekte des Projekts zu erzeugen, in definiertem Layout und mit Inhaltsverzeichnis.

3.16 Visualisierung

Information zur Visualisierung finden Sie im Abschnitt Visualisierungs-Editor.

3.17 Bibliotheksverwaltung

Bibliotheken können Funktionen, Funktionsbausteine, Datentypen, Globale Variablen und auch Visualisierungen bereitstellen, die dann im Projekt verwendet werden können wie direkt dort definierte Bausteine und Variablen.

Die Standard-Erweiterung für eine Bibliotheksdatei ist "**.library***, im Gegensatz zu ".lib", was für die Dateien in PLC Designer V2.3 und früheren Versionen verwendet wurden. Verschlüsselte Bibliotheken erhalten die Erweiterung ***.compiled-library**.

Die Verwaltung der Bibliotheken in einem Projekt erfolgt im Bibliotheksverwalter, die vorherige Installation auf dem System über den Bibliotheks-Repository-Dialog.

Die Projektfunktionen für globales und lokales Suchen & Ersetzen können auch für nicht verschlüsselte Bibliotheken verwendet werden.

Hinweis: Der Umfang der verfügbaren Funktionalität für die Bibliotheksverwaltung hängt vom aktuell verwendeten Feature-Set ab !

Installation auf dem System und Einbinden in ein Projekt

- Bibliotheken können auf dem lokalen System in verschiedenen "**Repositories**" (Verzeichnisse, Speicherorte) verwaltet werden. Bevor eine Bibliothek in ein Projekt eingebunden werden kann, muss sie auf dem lokalen System in ein solches Repository **installiert** werden. Dies erfolgt im **Bibliotheks-Repository Dialog** im PLC Designer.
- Als Voraussetzung für die Installation müssen die **Projektinformationen** eines Bibliotheksprojekts einen Titel, eine Versionsinfo und den Herstellernamen (Firma) erhalten. Optional kann eine Kategoriebezeichnung angegeben sein, die später im Bibliotheksverwalter zur Sortierung dienen kann.

- Wenn keine Kategoriezuweisung in den Projektinformationen vorliegt, gehört die Bibliothek automatisch zu Kategorie "Sonstige". Wenn neben dieser Standard-Kategorie weitere **Bibliothekskategorien** für im PLC Designer erstellte Bibliotheken verwendet werden, sind diese in einer oder mehreren externen XML-Datei(en) ***.libcat.xml** definiert, die auch erweitert bzw. neu erstellt werden können. Eine solche Datei kann dann im Projektinformationen-Dialog aufgerufen werden, um eine Kategorie auszuwählen.
- Bibliotheken werden mit Hilfe des **Bibliotheksverwalters** in ein Projekt **eingebunden**. In einem "Standardprojekt" ist dieser zunächst automatisch dem Standard-Gerät zugewiesen. Er kann aber auch explizit im Gerätebaum (unterhalb eines Geräts oder einer Applikation) oder im POUs-Fenster hinzugefügt werden. Dies erfolgt wie für andere Objekte über den Objekt hinzufügen Dialog. Bibliotheken, die in einer Bibliothek eingebunden sind, werden standardmäßig im Bibliotheksverwalter ebenfalls dargestellt, "versteckte Bibliotheken" sind allerdings auch möglich, siehe Referenzierte Bibliotheken, "Beim Erstellen... 2.".
- Liegt die Bibliothek nicht nur in verschlüsselter Form vor, sondern ist auch die "**.library***" Datei vorhanden, so können die im Bibliotheksverwalter aufgeführten POUs der Bibliothek durch einen Doppelklick auf den jeweiligen Eintrag geöffnet werden.
- Wenn eine Bibliotheksfunktion im Projekt angesprochen wird, werden die Bibliotheken und Repositories in der **Reihenfolge** durchsucht, in der sie im Bibliotheks-Repository Dialog aufgeführt sind. Sehen Sie unten zur Eindeutigkeit von Zugriffen.

Referenzierte Bibliotheken

- Eine Bibliothek kann andere Bibliotheken einbinden (**referenzierte Bibliotheken**), wobei die Verschachtelung beliebig tief sein kann. Wenn eine solche "Vater"-Bibliothek dann selbst in ein Projekt eingebunden wird, stehen die in ihr referenzierten Bibliotheken dort ebenfalls zur Verfügung. Achtung: Allerdings nur diejenigen, die in der "Vater"-Bibliothek im globalen Bibliotheksverwalter (POUs-Fenster) eingebunden sind!
- **Beim Erstellen** eines Bibliotheksprojekts, das andere referenziert, kann in den Eigenschaften jeder eingebundenen Bibliothek festgelegt werden, wie sie sich später, wenn sie über die "Vater"-Bibliothek in ein Projekt eingebunden wird, verhalten soll:
  - 1. Ihre Sichtbarkeit im Bibliotheksverwalter, eingerückt unterhalb der "Vater"-Bibliothek, kann deaktiviert werden. Somit können **"versteckte Bibliotheken"** in einem Projekt bereitgestellt werden.

  - 2. Wenn gerade eine reine **"Container"-Bibliothek** erzeugt wird, also ein Bibliotheksprojekt, das selbst keine Bausteine definiert, sondern nur andere Bibliotheken referenziert, kann der spätere Zugriff auf deren Bausteine vereinfacht werden. Eine "Container"-Bibliothek erstellt man, um mit ihrem Einbinden in ein Projekt gleichzeitig ein ganzes Set von Bibliotheken einbinden zu können. In diesem Fall ist es aber dann ev. gewünscht, den Zugriff auf die Bausteine dieser Bibliotheken zu vereinfachen, indem sie zu "Top-Level"-Bibliotheken erklärt werden und daraufhin beim Zugriff auf die Bausteine ggfs. der Namensraum der Bibliotheken weggelassen werden kann. Dazu dient die Option "Publish..." in den Bibliothekseigenschaften. Diese Option sollte allerdings **NUR beim Erstellen eines Container-Bibliotheksprojekts** und mit Bedacht verwendet werden!
- Zum Thema "eindeutiger Zugriff auf referenzierte Bibliotheken" sehen Sie unten.

Bibliotheksversionen

- Mehrere Versionen einer Bibliothek können gleichzeitig auf dem System installiert sein.
- Mehrere Versionen einer Bibliothek können gleichzeitig im Projekt eingebunden sein. Folgendermaßen wird eindeutig geregelt, auf welche Version eine Applikation in diesem Fall zugreift:

  Wenn mehrere Versionen auf derselben Ebene im Bibliotheksverwalter vorliegen, hängt es von der Definition in den Bibliotheks-Eigenschaften ab, welche Version verwendet wird (eine bestimmte oder immer die neueste).

  Wenn mehrere Versionen auf verschiedenen Ebenen vorliegen (was im Falle von referenzierten Bibliotheken der Fall sein kann), wird ein eindeutiger Zugriff auf Bibliotheksbausteine durch die Angabe des entsprechenden **Namensraums** erreicht; siehe folgender Abschnitt.

Eindeutiger Zugriff auf Bibliotheksbausteine (Namensraum)

- Grundsätzlich gilt folgendes: Wenn im Projekt mehrere Bausteine gleichen Namens verfügbar sind, muss der Zugriff auf eine Bausteinkomponente eindeutig sein, ansonsten treten Compiler-Fehler auf. Dies gilt sowohl für projektlokale Bausteine wie auch für solche, die in eingebundenen und in diesen referenzierten Bibliotheken bereitstehen. Die Eindeutigkeit wird in solchen Fällen durch Hinzufügen des Namensraums vor dem Bausteinnamen erreicht.
- Der **Standard-Namensraum** einer Bibliothek ist in den Bibliotheks-Eigenschaften definiert. Wird er nicht explizit definiert, wird automatisch der Bibliotheksname dafür verwendet, es kann aber auch beim Erstellen eines Bibliotheksprojekts im Eigenschaften-Dialog ein anderer Standard-Namensraum eingetragen werden. Später, wenn eine Bibliothek bereits im Bibliotheksverwalter eines Projekts eingebunden ist, kann der Namensraum jederzeit lokal durch den Anwender - ebenfalls im Eigenschaften-Dialog - verändert werden.

- Beispiele: In den folgenden Beispielen sei der "Namensraum" der Bibliothek *Lib1* in den Bibliothekseigenschaften mit "Lib1" eingetragen. Sehen Sie in der rechten Spalte die Verwendung der Namensräume für eindeutige Zugriffe auf Variable *var 1*, die in Baustein *module1* und in Baustein *POU1* definiert ist.

	Variable var1 ist an allen Stellen (1) bis (5) im Projekt definiert:	Eindeutiger Zugriff auf var1 mit Hilfe der entsprechenden Namensraum-Angaben...
(1)	In Bibliothek *Lib1* im globalen Bibliotheksverwalter im POUs-Fenster	"Lib1.module1.var1"
(2)	In Bibliothek *Lib1* im Bibliotheksverwalter unterhalb von Applikation *App1* von Gerät *Dev1* im Geräte-Fenster	"Dev1.App1.Lib1.module1.var1"
(3)	In Bibliothek *Lib1*, die in Bibliothek *F_Lib* eingebunden (**referenziert**) ist, im globalen Bibliotheksverwalter im POUs-Fenster	Standardmäßig: (Option "Publish..." ist beim Einfügen von Lib1 in F_Lib in den Bibliothekseigenschaften von Lib1 deaktiviert): "F_Lib.Lib1.module1.var1" Wenn Option "Publish..." aktiviert wäre, würdemodule1 wie eine Komponente einer top-level eingebundenen Bibliothek behandelt. Dann wäre normalerweise auch ein Zugriff ohne Angabe des Namensraums der "Vater"-Bibliothek F_Lib möglich: "Lib1.module1.var1" oder "module1.var1"). Im vorliegenden Beispiel würde dies allerdings zu einem Compiler-Fehler führen, da der Aufruf dann nicht mehr eindeutig wäre, siehe (1) und (4).
(4)	Im Objekt *module1* das im POUs-Fenster definiert ist	"module1.var1"
(5)	Im Objekt *POU1* das im POUs-Fenster definiert ist	"POU1.var1"

Erstellen von V3.x Bibliotheken, Verschlüsseln, Dokumentieren

Ein PLC Designer-Projekt kann als Bibliothek gespeichert werden (<projektname>.library) und optional auch gleichzeitig im "System"-Bibliotheks-Repository installiert werden. Für eine Bibliothek werden dabei nur die im POUs-Fenster definierten Bausteine berücksichtigt. Wenn gezielt ein Bibliotheksprojekt erstellt werden soll, ist zu empfehlen, im Dialog 'Neues Projekt' direkt die Vorlage "Leere Bibliothek" auszuwählen.

Beim Erstellen muss folgendes beachtet werden:

- In den Projektinformationen müssen ein **Titel**, eine **Versionsangabe** und die **Firma** eingetragen werden. Wenn der **Standard-Namensraum** anders lauten soll als der Bibliotheksname, kann dies gleich hier definiert werden. Optional ist auch die Angabe einer **Kategorie** zu empfehlen, da diese später im Bibliotheks-Repository Dialog und im Bibliotheksverwalter zur Sortierung dient. Wenn die Bibliothek nicht zur Standard-Kategorie "Sonstige" gehören soll, muss eine entsprechende Kategorien-Beschreibung geladen werden, entweder aus einer XML-Datei ***.libcat.xml** oder aus einer anderen Bibliothek, die bereits die Informationen einer solchen Beschreibungsdatei beinhaltet. Gegebenenfalls muss die Kategorien-Beschreibungsdatei neu erstellt oder eine bestehende modifiziert werden. Der Inhalt der Beschreibungsdatei wird ins Bibliotheksprojekt übertragen und dann bei der Installation mit dieser ins Bibliotheks-Repository. Daraufhin sind die Kategorien im Repository bekannt. Wenn danach eine weitere Bibliothek bei ihrer Installation eine Beschreibungsdatei derselben Kennung, aber mit verändertem Inhalt "mitbringt", gilt im gesamten Repository für die Anzeige der Kategorien die Information aus der neueren Datei.
- Wenn die Bibliothek andere Bibliotheken einbindet, sollte überlegt werden, wie sich diese **referenzierten Bibliotheken** später, wenn die "Vater"-Bibliothek in ein Projekt eingebunden wird, verhalten sollten. Dabei geht es um Versionshandhabung, Namensraum, Sichtbarkeit und Zugriffsmöglichkeiten, die im Eigenschaften-Dialog der einzelnen referenzierten Bibliothek konfiguriert werden. Wenn die Bibliothek später bei der Verwendung je nach **Zielsystem** eine andere Bibliothek referenzieren soll, können jetzt beim Definieren der Referenz **Platzhalter** eingesetzt werden.
- Wenn die Bibliotheksbausteine vor Ansicht und Zugriff geschützt werden sollen, kann ein Bibliotheksprojekt in **verschlüsseltem Format** gespeichert werden (<projektname>.compiled-library).
- Datenstrukturen einer Bibliothek können als Bibliotheks-intern markiert werden. Diese nicht-öffentlichen Objekte tragen das Attribut 'hide' und erscheinen daher nicht innerhalb der Bibliotheksverwaltung, der "Komponenten Auflisten" Funktionalität oder des Eingabe-Assistenten.

- Um dem Anwender der Bibliothek auf einfache Weise Informationen zu einem Baustein bereitzustellen, kann beim Erstellen eines Bibliotheksprojekts der Deklaration eines Bausteinparameters ein entsprechender Kommentare hinzugefügt werden. Dieser wird dann später, wenn die Bibliothek in ein Projekt eingebunden ist, im Bibliotheksverwalter auf einem Registerblatt '**Dokumentation**' dargestellt.
- Die folgenden Befehle stehen standardmäßig im Datei-Menü zum Speichern eines Bibliotheksprojekts zur Verfügung:
  - Projekt speichern als...
  - Projekt speichern und ins Bibliotheks-Repository installieren
  - Projekt als kompilierte Bibliothek speichern (Verschlüsseln)

PLC Designer V2.3 Bibliotheken

- Bibliotheken, die mit PLC Designer V2.3 (*.lib) und früheren Versionen erstellt wurden, werden weiterhin unterstützt:
- Ein "altes" Bibliotheksprojekt (*.lib) kann in PLC Designer V3.x direkt geöffnet und dabei in eine "V3.x Bibliothek" (*.library) konvertiert werden.
- Wenn ein altes Projekt geöffnet wird, das alte Bibliotheken referenziert, kann gewählt werden, ob diese Referenzen beibehalten, durch andere ersetzt oder gelöscht werden sollen. Sollen sie beibehalten werden, werden die betreffenden Bibliotheken in das neue Format konvertiert und dabei automatisch in das System-Bibliotheks-Repository installiert. Falls sie die nötigen Projektinformationen nicht enthalten, können diese unmittelbar nachgetragen werden. Das Muster (Mapping), nach dem eine alte Bibliothek einmal beim Konvertieren eines alten Projektes gehandhabt wurde, kann in den Projekt-Optionen gespeichert werden, so dass dieselbe Bibliothek bei künftigen Projektkonvertierungen nicht jedesmal erneut explizit behandelt werden muss.

Externe und interne Bibliotheken bzw. Bibliotheksbausteine, spätes Binden

- Eine "**externe Bibliothek**" ist im Unterschied zu einer internen (PLC Designer Bibliotheksprojekt) eine Bibliotheksdatei, die außerhalb vom PLC Designer in einer anderen Programmiersprache, wie beispielsweise C, programmiert wird. Sie muss auf dem Zielsystem vorliegen und wird erst gebunden, wenn die Applikation dort läuft.
- Um die Erstellung einer externen Bibliothek vorzubereiten, erlaubt der PLC Designer **die entsprechenden C-Frame und C-Schnittstellen-Dateien** (m4-Dateiformat) aus dem aktuellen PLC Designer-Bibliotheksprojekt zu generieren. Die Verwendung dieser Funktionalität garantiert die Konsistenz zwischen der PLC Designer-Bibliothek und ihrer externen Implementierung. Sehen Sie hierzu den entsprechenden Befehl 'Erzeuge Laufzeitsystemdateien'.
- Wie in V2.3 ist es auch möglich, eine PLC Designer-Bibliothek als externe Bibliothek **später zu binden**, d. h. erst wenn die Applikation auf dem Laufzeitsystem betrieben wird. Zusätzlich ist es nun sogar möglich, das "späte Binden" für jeden Bibliotheksbaustein einzeln zu definieren. Zu diesem Zweck kann die Eigenschaft in den Objekteigenschaften ("Externe Implementierung") eines oder aller Bausteine aktiviert werden.

3.18 Sicherheit

Projekt:

Die Kontrolle des Zugriffs auf Projekte, der Rechte bestimmte Aktionen auf einzelne Objekte bzw. in einem Projekt auszuführen kann in den Dialogen der Projekteinstellungen und Objekteigenschaften konfiguriert werden.

Ein Projekt kann grundsätzlich mit einem **verschlüsselten Passwort** geschützt werden, siehe Projekteinstellungen, Sicherheit.

Zugriffsrechte für Objekte werden immer einer Benutzergruppe zugewiesen, nicht einzelnen Benutzern. Für jedes Objekt kann für jede Benutzergruppe eine Liste erlaubter Aktionen definiert werden.

Jeder einzelne Benutzer kann ein eigenes **Passwort** erhalten.

Steuerung:

Abhängig vom Gerät ist ggfs. auch ein Zugriffsschutz für Objekte und Dateien auf der Steuerung konfigurierbar.

4 Quickstart

4.1 Hilfe

Derzeit ist eine nicht-dynamische Version der Online Hilfe implementiert. Standardmäßig kann über das 'Hilfe'-Menü darauf zugegriffen werden. Die Sprache, in der die Hilfeseiten angezeigt werden, kann im Dialog Optionen / Internationale Einstellungen umgestellt werden.

Kontextsensitive Hilfe

Kurzform: <F1>

Sie können die Taste <F1> in einem aktiven Fenster, einem Dialog oder über einem Menübefehl betätigen, um die Online Hilfe zu öffnen. Wenn ein Menübefehl angewählt ist, wird unmittelbar die Hilfeseite für diesen Befehl angezeigt.

Ebenso, wenn Sie einen Text markieren (z. B. ein Schlüsselwort, eine Standardfunktion oder eine Fehlermeldung im Meldungsfenster), wird über <F1> die entsprechende Hilfe dazu angezeigt.

4.2 Deinstallation, Aktualisieren, Reparieren

Um das Entwicklungssystem und seine Komponenten zu deinstallieren, oder um die aktuelle Installation zu verändern, führen Sie bitte die aktuelle Setup-Datei aus. Sie werden durch entsprechende Dialoge geleitet.

4.3 Projekt erstellen und auf der Steuerung betreiben

Sehen Sie im folgenden eine Beschreibung wie ein einfaches Projekt erstellt wird, das ein Steuerungsprogramm enthält, und wie dieses Programm als "Applikation" auf die Steuerung gebracht wird (Gateway-Verbindung) und dort gestartet und kontrolliert wird. Das im Beispiel verwendete Steuerungs-Laufzeitsystem wird mit dem Standard-Setup des PLC Designer-Entwicklungssystems installiert.

Das Beispielprogramm wird in Strukturiertem Text erstellt und besteht aus einem Programm *PLC_PRG* und einem Funktionsblock *FB1*. *PLC_PRG* enthält eine Zählervariable *ivar* und ruft Funktionsblock *FB1* auf; *FB1* erhält auf dem Eingang "in" einen Wert von *PLC_PRG*, addiert diesem "2" hinzu und stellt das Ergebnis in Ausgang *out* bereit, *out* wird wiederum von *PLC_PRG* gelesen.

Beachten sie, dass sich die folgende Beschreibung auf die Standard-Konfiguration der Benutzeroberfläche bezieht, die nach einer Standard-Installation des Programmiersystems vorliegt.

Hinweis: Wenn Ihr System PLC Designer-SoftMotion unterstützt, beachten Sie die Beschreibung einiger Beispielprojekte auf den Hilfeseiten zum SoftMotion Geräteeditor.

PLC Designer starten und Projekt anlegen

(1) PLC Designer starten

Wählen Sie im Start-Menü Ihres Rechners folgendes:

Programme > **Lenze** > **PLC Designer** > **PLC DesignerV3.x**

Alternativ kann über das PLC Designer-Symbol gestartet werden, das nach der Installation auf dem Desktop verfügbar ist.

(Künftig: Sie werden zur Auswahl eines Profils aufgefordert und nach Drücken von **Weiter** wird die PLC Designer-Benutzeroberfläche geöffnet.)

Das Programmiersystem wird automatisch mit einem vordefinierten Profil gestartet, d. h. derzeit muss vom Anwender noch kein Profil eingestellt werden.

Wenn das Programmiersystem das erste Mal nach der Erstinstallation auf dem System gestartet wird, wird der Anwender aufgefordert, die Standardauswahl an Funktionalität für die Programmierumgebung festzulegen.

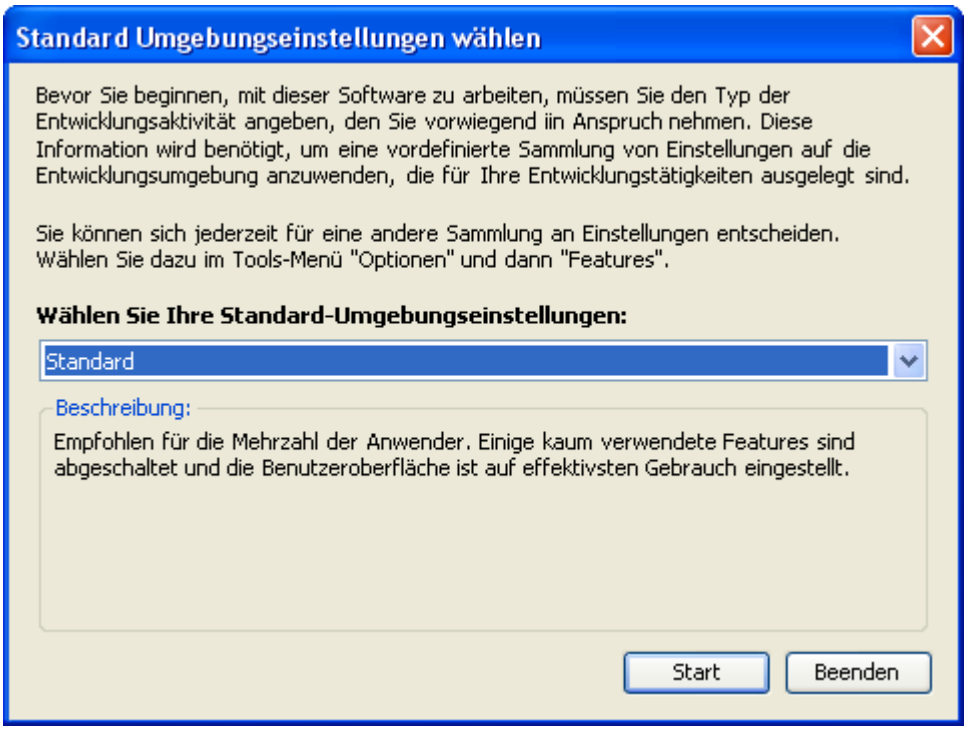
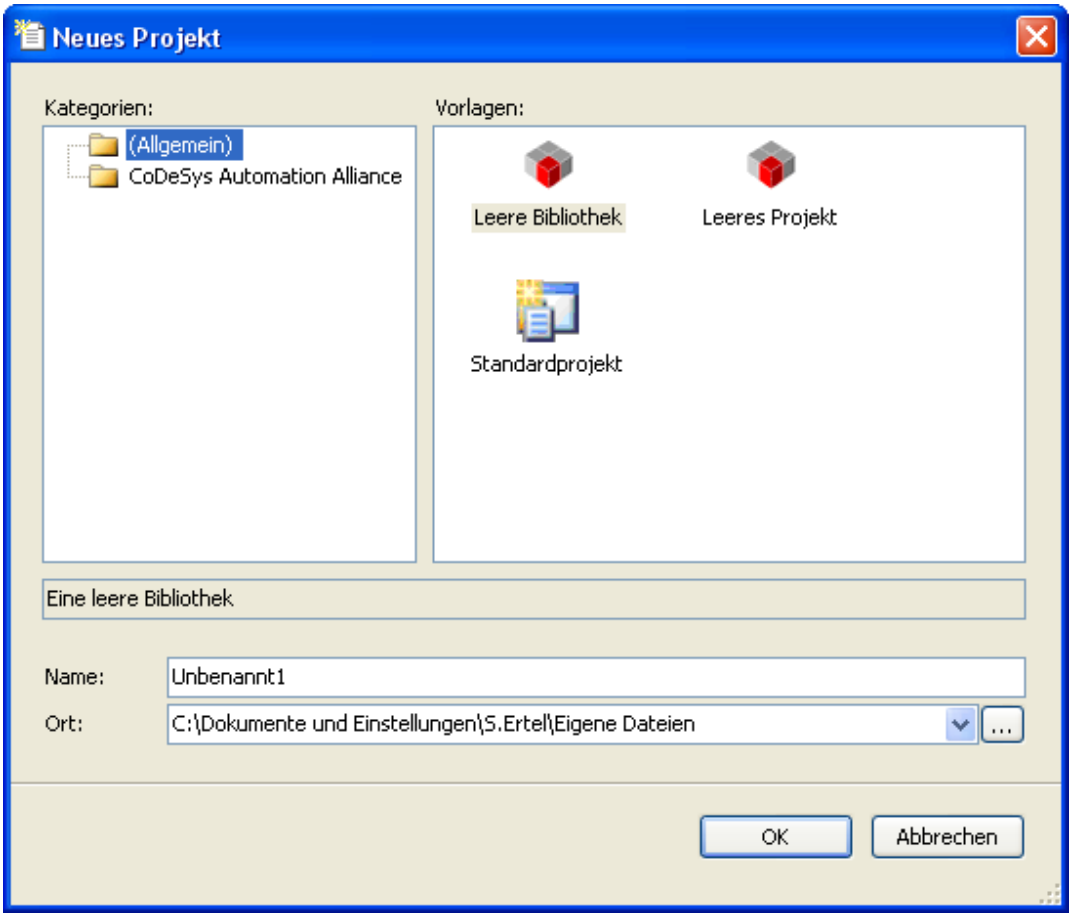


Abb. 38 Dialog 'Standard hoose Default Environment Settings'

Wählen Sie "Standard" für die Ersten Schritte, die hier in der Hilfe beschrieben sind, Sie können später immer noch auf die "Professional" Einstellung wechseln (über den Projekt Optionen Dialog). In der "Standard" Umgebung ist die Benutzeroberfläche auf effektivste Bedienbarkeit ausgelegt und einige selten verwendete Features sind ausgeschaltet.

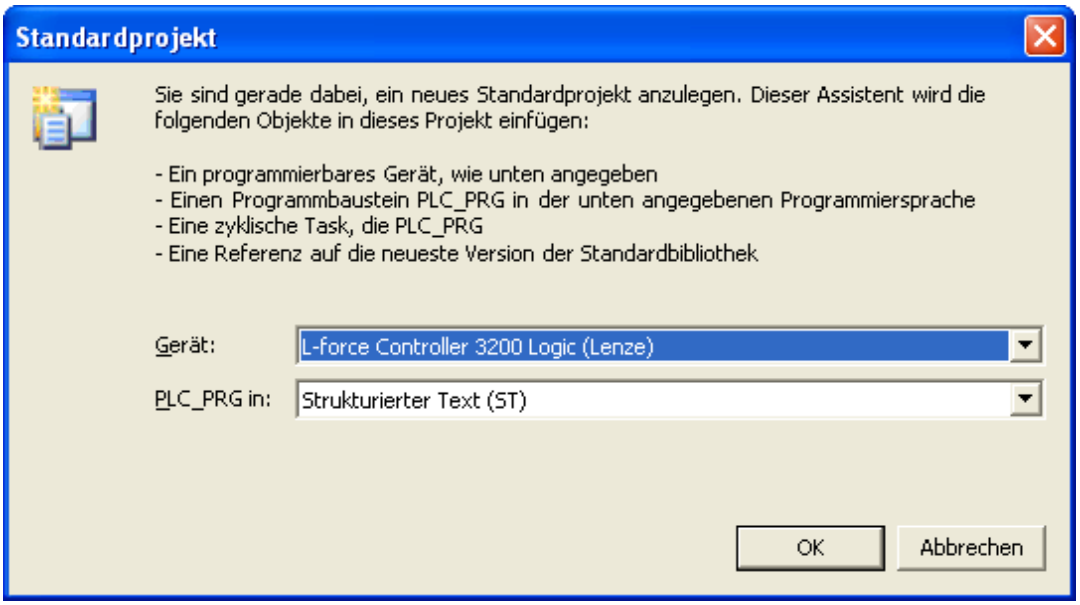
(2) Projekt anlegen

Um ein neues Projekt,anzulegen, wählen sie den Befehl **Neues Projekt** im **Datei**-Menü:



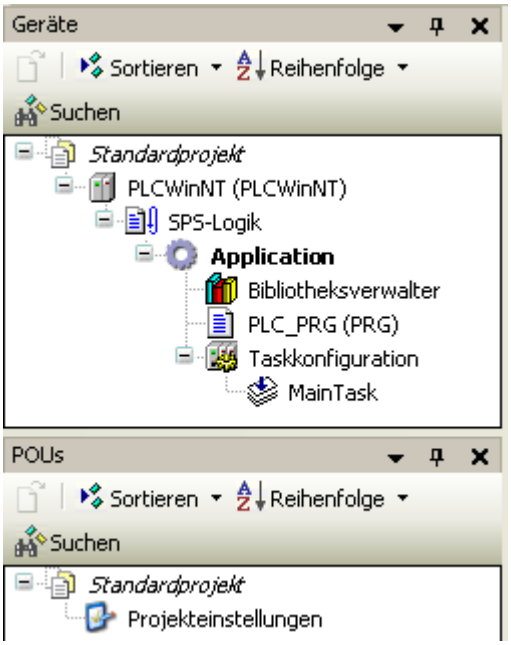
Im Dialog **Neues Projekt** wählen sie im Feld 'Vorlagen' die Vorlage **Standardprojekt** und geben einen **Namen** und bei **Ort** einen Dateipfad für die Projektdatei ein. Bestätigen Sie mit **OK**.

Es erscheint ein Wizard-Dialog:



Wählen Sie das Gerät L-force Controller 3200 Logic (Lenze) or L-force Controller 3200 Motion (Lenze) und die Programmiersprache Strukturierter Text für PLC_PRG. Bestätigen Sie mit **OK**.

Der Projektname erscheint nun in der Titelleiste der PLC Designer-Benutzeroberfläche und als symbolischer Wurzelknoten im POUs- wie auch im Gerätefenster.



Das POUs-Fenster enthält die Projekteinstellungen.

Das Gerätefenster zeigt eine Baumstruktur, welche das Gerät "Device (L-force Controller 3200)" mit einer unterhalb eingefügten Applikation abbildet. Letztere beinhaltet das Programm (in Strukturiertem Text editierten) "PLC_PRG" und die obligatorische Taskkonfiguration mit einer "MainTask" zur Steuerung von PLC_PRG.

Weiterhin ist bereits ein Bibliotheksverwalter eingehängt, der automatisch die Bibliothek "IoStandard.library" enthält, die für E/A-Konfigurationen benötigt wird, sowie die "Standard.library", die alle Funktionen und Funktionsbausteine bereitstellt, die gemäß der Norm IEC61131-3 als Standardbausteine für ein IEC-Programmiersystem benötigt werden .

(Der zusätzliche Knoten "SPS-Logik" unterhalb des Knotens "Device (L-force Controller 3200)" ist nur ein symbolischer Knoten, der anzeigt, dass das Gerät "programmierbar" ist.)

Der symbolische Gerätename kann umbenannt werden, indem der Eintrag ausgewählt, mit der <Leertaste> ein Editierfeld geöffnet, und anstelle von "Device" ein anderer Name eingegeben wird. Ändern Sie den Namen ab in "**PLCWinNT**".

Ein Steuerungsprogramm schreiben

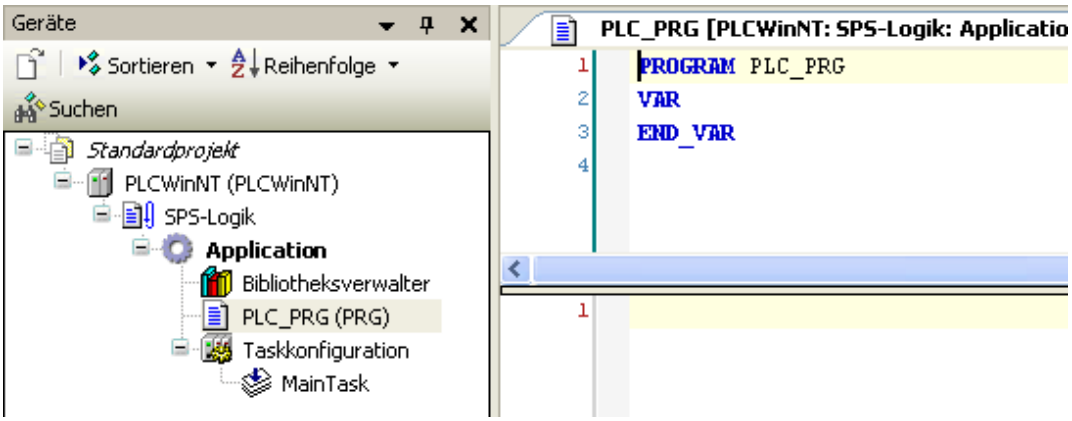
(3) Variablen deklarieren in "PLC_PRG"

Öffnen Sie den Baustein "PLC_PRG", der standardmäßig bereits im Geräte-Fenster eingefügt ist, im **ST (Strukturierter Text) Editor**. Dies geschieht über einen Doppelklick auf den Eintrag "PLC_PRG" im Gerätebaum.

Der Editor besteht aus einem Deklarationsteil (oben) und einem Implementierungsteil (unten), getrennt durch einen verschiebbaren Balken.

Der Deklarationsteil enthält nummerierte Zeilen, in der ersten den Bausteintyp und -namen ("PROGRAM PLC_PRG") und in den weiteren die Schlüsselwörter "VAR" und "END_VAR", die die Variablendeklaration einklammern werden.

Der Implementierungsteil ist noch leer, nur Zeilennummer 1 wird angezeigt:



Stellen Sie nun den Cursor Im Deklarationsteil hinter "VAR" und drücken Sie die Eingabetaste, woraufhin eine Leerzeile eingefügt wird. In dieser geben Sie die Deklarationen der Variablen ein: "ivar" und "erg" vom Typ INT, sowie "fbinst" vom Typ "FB1".

```
PROGRAM PLC_PRG
VAR
ivar: INT;
fbinst: FB1;
erg: INT;
END_VAR
```

Alternativ können Sie direkt im Implementierungsteil Codezeilen eingeben und zur Deklaration der darin verwendeten Variablen die Autodeklarationsfunktion verwenden.

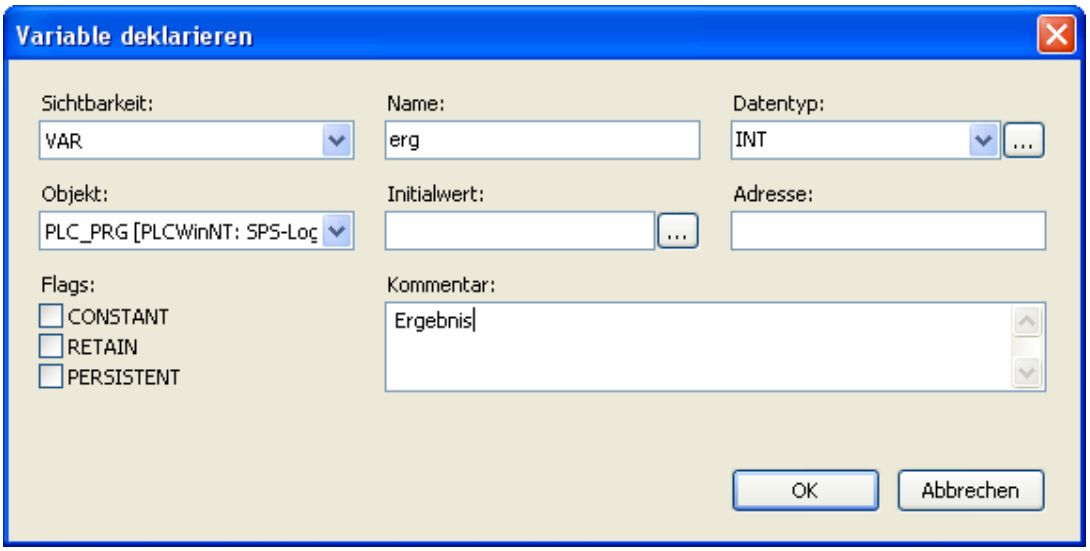
(4) Programm-Code im Implementierungsteil von "PLC_PRG" eingeben

Setzen Sie den Cursor in Zeile 1 im Implementierungsteil von PLC_PRG und geben Sie folgende Code-Zeilen ein:

```
ivar := ivar+1;          // counter
fbinst(in:=11, out=>erg);   // call of function block of type FB1
// with input parameter "in"
// output is written to "erg"
```

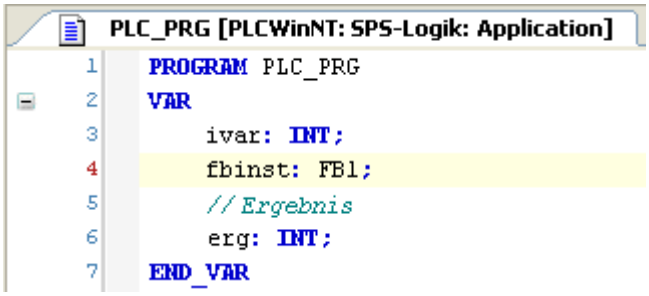
Für Variable *in* wird automatisch der Dialog 'Variable deklarieren' geöffnet, bitte mit Abbrechen schließen (wird später im Funktionsblock FB1 behandelt).

Anstelle der Schritte (3) und (4) kann auch die Autodeklarations-Funktion verwendet werden. Geben Sie - ohne vorher Variablen zu deklarieren - im Implementierungsteil des Editors die erste Code-Zeile ein und drücken dann die Eingabetaste. Für jede noch nicht deklarierte Variable der Zeile wird der Dialog 'Variable deklarieren' geöffnet, wo die Deklaration vorgenommen werden kann:



Der Variablenname und sein Gültigkeitsbereich ("Sichtbarkeit"), wie auch der aktuelle Baustein (Objekt) werden automatisch vorgegeben. Geben Sie den gewünschten Datentyp und Initialwert entsprechend der in (3) beschriebenen Deklaration ein und fügen Sie den Kommentar wie im Dialogbild gezeigt hinzu.

Bestätigen Sie den Deklarationsdialog mit OK. Daraufhin wird die Deklaration von ivar im Deklarationsfenster eingefügt. Der Kommentar wird oberhalb der Deklarationszeile dargestellt:



(5) Eine weitere POU anlegen (ST Funktionsblock "FB1")

Nun fügen wir einen Funktionsblock hinzu, welcher zu dem in der Eingangsvariablen "in" enthaltenen Wert 2 addiert und das Ergebnis in der Ausgangsvariablen "out" bereitstellt:

Wählen Sie Befehl **Objekt hinzufügen** im Menü **Projekt**.

Wählen Sie 'POU' im linken Teil des Dialogs 'Objekt hinzufügen'. Geben Sie den Namen "FB1" ein, aktivieren Sie bei "Typ" die Option **Funktionsblock**.

Wählen Sie 'Strukturierter Text (ST)' als **Implementationssprache**.

Betätigen Sie Schaltfläche **Öffnen** um die Objekteinstellungen zu bestätigen. Der Funktionsblock wird als neuer Baustein im POUs Fenster angelegt und ist somit (im Unterschied zu PLC_PRG) keiner bestimmten Applikation zugeordnet.

Zusätzlich öffnet sich ein weiteres Editor-Fenster, in dem der Funktionsblock FB1 bearbeitet werden kann. Deklarieren Sie dort in der gleichen Weise wie schon für PLC_PRG beschrieben die folgenden Variablen:

```
FUNCTION_BLOCK FB1
VAR_INPUT
 in:INT;
END_VAR
VAR_OUTPUT
 out:INT;
END_VAR
VAR
 ivar:INT:=2;
END_VAR
```

Im Implementierungsteil des FB1-Fensters geben Sie folgendes ein:

```
out:=in+ivar;
```

Definieren der Resourcen-Objekte für das Betreiben und Kontrollieren des Programms auf der Steuerung

(6) Gateway Server and PLC starten

Gateway Server starten:

Der Gateway Server wird beim Systemstart automatisch als Dienst gestartet. Prüfen Sie, ob das Symbol in der Systemleiste am unteren Rand des Bildschirms erscheint. Wenn das Symbol so erscheint: , ist der Gateway gestoppt.

(Das oben beschriebene Symbol ist Bestandteil des GatewaySysTray Programms, das zur Kontrolle und Anzeige des Gateway-Dienstes dient. Es bietet ein Menü mit Start- und Stop-Befehlen, worüber der Dienst jederzeit vom Anwender händisch gestoppt und neu gestartet werden kann.

NOCH NICHT VERFÜGBAR: Gateway "Inspector"-Funktion.

Das Menü enthält auch den Befehl "Exit Gateway Control", der nur das GatewaySysTray Programm, nicht aber den Gateway-Dienst beendet. Das GatewaySysTray Programm wird automatisch gestartet, wenn Windows gestartet wird, kann jedoch auch manuell über das Programme-Menü gestartet werden).

Steuerung starten:

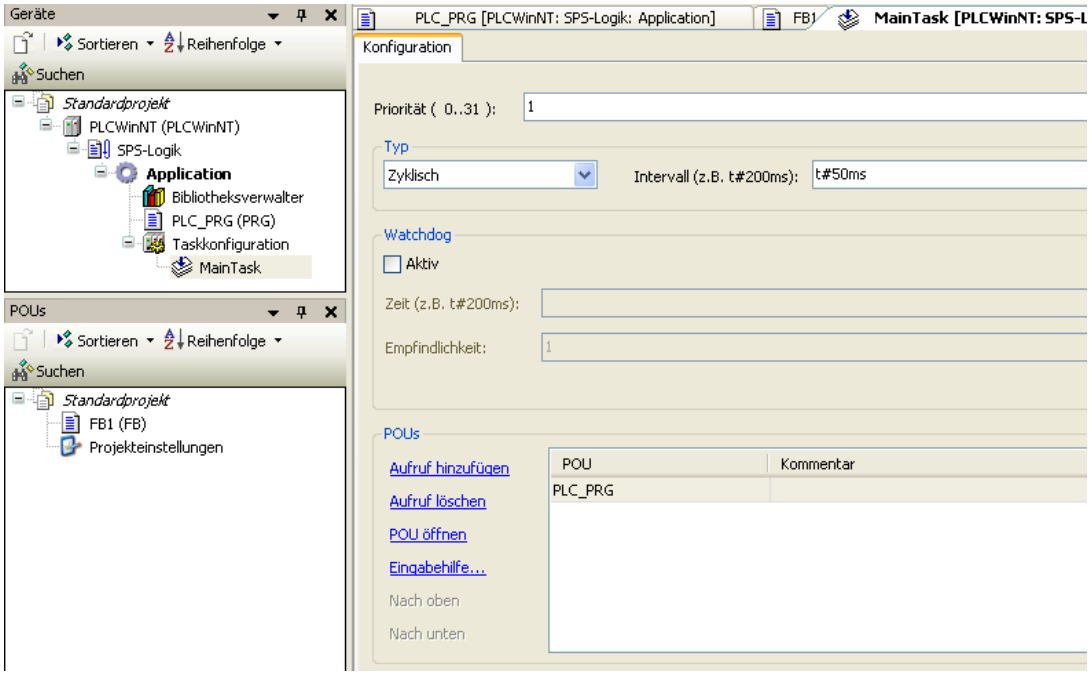
Die Steuerung (L-force Controller 3200) ist ab Systemstart als Dienst verfügbar. Sie wird wie der Gateway durch ein Symbol in der Systemleiste vertreten: für Status 'gestoppt', für Status 'läuft'. Wenn es das System erlaubt, wird der Steuerungs-Dienst automatisch beim Systemstart mit gestartet. Ansonsten muss er manuell über den Befehl "Start PLC" gestartet werden, der in dem Menü zu finden ist, das sich bei einem Mausklick auf das Symbol öffnet.

(Das Steuerungssymbol ist Bestandteil des PLCDesignerSPSysTray Progrmms, das zur Kontrolle und Anzeige des -force Controller 3200-Dienstes dient. Es bietet ein Menü mit Start- und Stop-Befehlen, worüber der Dienst jederzeit vom Anwender händisch gestoppt und neu gestartet werden kann.

Das Menü enthält auch den Befehl "Exit PLC Control", der nur das PLCDesignerSPSysTray Programm, nicht aber den Steuerungs-Dienst beendet. Das PLCDesignerSPSysTray Programm wird automatisch gestartet, wenn Windows gestartet wird, kann jedoch auch manuell über das Programme-Menü gestartet werden).

(7) "Aktive Applikation" setzen

Bei einem Klick auf 'MainTask' im Geräte-Fenster des "Standardprojekts" öffnet sich ein Fenster, das die Konfiguration der Task enthält:



Der Name der Applikation '**Application**'ist im Geräte-Fenster fett dargestellt. Das bedeutet, dass diese Applikation als "aktive" Applikation gesetzt wurde. Damit bezieht sich alle Kommunikation mit der Steuerung auf diese Applikation.

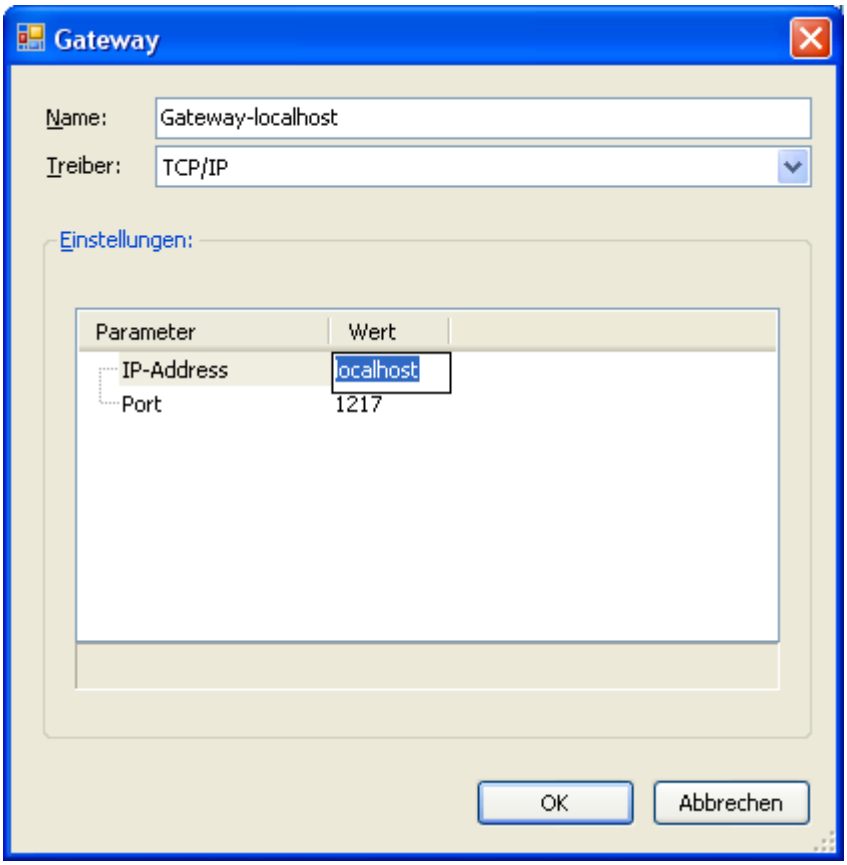
Um eine Applikation als aktive Applikation zu setzen wählen Sie im Geräte-Fenster den Eintrag 'Application' mit der linken Maustaste aus. Bei einem anschließenden Klick auf die rechte Maustaste erscheint ein Kontextmenü, das den Befehl **Aktive Applikation setzen** enthält.

(8) Kommunikationskanal zur Steuerung konfigurieren

Führen Sie einen Doppelklick auf den Eintrag "Device (L-force Controller 3200)" im Geräte-Fenster aus. Der Dialog "PLCWintNT" öffnet mit Unterdialog "Kommunikationseinstellungen". Hier muss die Verbindung zwischen Steuerung (Gerät) und Programmiersystem konfiguriert werden. Sehen Sie dazu die folgenden Schritte, der eingerichtete Kanal muss schließlich in der Zeile bei 'Netzwerkpfad zum Gerät auswählen' definiert sein:

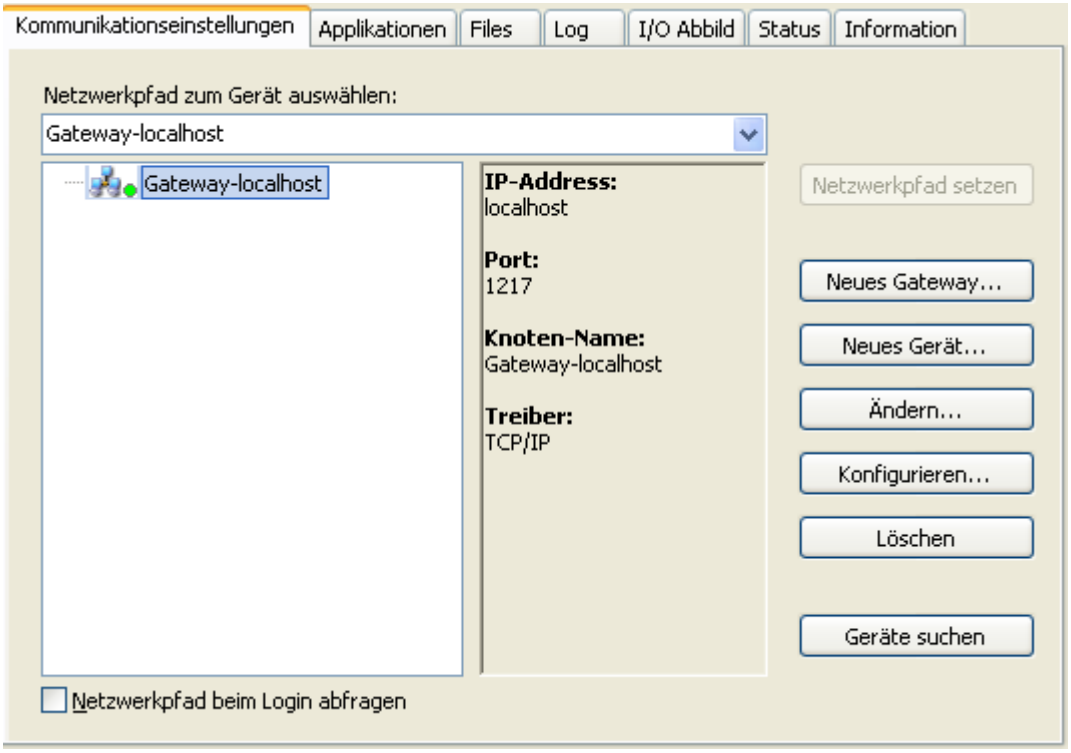
Sofern dies Ihre erste Kommunikationskonfiguration mit PLC Designer V3.x ist, müssen Sie nun den lokalen Gateway Server definieren. (Wenn dies vorher schon einmal gemacht wurde, wird er nun bereits im Kommunikationseinstellungen-Dialog angezeigt, wie im übernächsten Bild zu sehen ist. In diesem Fall können Sie nun gleich zur Definition des Kommunikationskanals zum Gerät weiter springen.)

Der Gateway Server wird mit dem PLC Designer-Setup bereitgestellt. Betätigen Sie Schaltfläche **Neues Gateway** um folgenden Gateway-Dialog zu öffnen:



Geben Sie einen symbolischen **Namen** für den Gateway ein, stellen Sie als Treibertyp "TCP/IP" ein und geben Sie die IP-Adresse "localhost" an. (Wählen Sie dazu das Feld in Spalte "Wert" aus und öffnen mit der Leertaste ein Eingabefeld.) Belassen Sie die Einstellung für "Port" und bestätigen Sie mit **OK**.

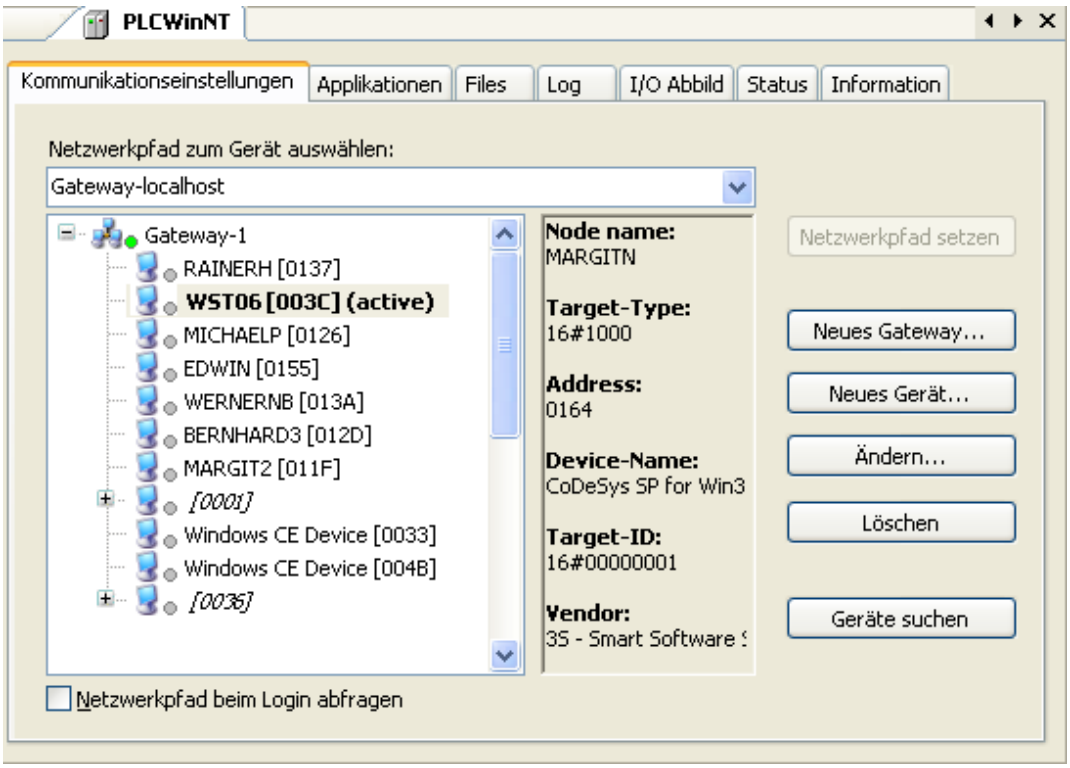
Der Gateway wird nun im linken Feld des Kommunikationseinstellungen-Dialogs eingetragen und der Auswahlliste bei 'Netzwerkpfad zum Gerät auswählen' hinzugefügt. Wenn der Gateway korrekt läuft, wird ein grün gefüllter Kreis am Symbol des Eintrags angezeigt, ansonsten ein roter.



Nun definieren Sie den Kommunikationskanal zum Gerät, welcher dann über den gesetzten Gateway verwendet wird.

Betätigen Sie die Schaltfläche **Geräte suchen**, um das lokale Netzwerk nach allen verfügbaren Geräten abzusuchen.

Zumindest die mit dem PLC Designer-Setup automatisch installierte Steuerung sollte gefunden werden; sie wird im Fenster eingerückt unterhalb des Gateway-Eintrags angezeigt. Sehen Sie hierzu die folgende Abbildung. Anstelle von "WST06 [003C]" steht allerdings die Adresse Ihres Computers. Wenn die Steuerung nicht gefunden wird, überprüfen Sie, ob sie läuft, siehe (6).



Wählen Sie nun den Geräteeintrag aus und betätigen Sie die Schaltfläche **Netzwerkpfad setzen**.

Dies wird den Kanal aktiv setzen, was bedeutet, dass alle Kommunikationsaktionen sich genau auf ihn beziehen. Beachten Sie dies später im Falle, dass Sie mehrere Kommunikationskanäle im Projekt haben.

Schließen Sie den Kommunikationseinstellungen-Dialog mit **OK** um die Einstellungen wirksam zu machen.

Die Applikation auf der Steuerung starten und kontrollieren

(9) Applikation übersetzen und auf die Steuerung laden

Soll lediglich eine syntaktische Prüfung der "aktiven" Applikation durchgeführt werden, so verwendet man den Befehl "'<Application...>' übersetzen" (Kontextmenü oder Übersetzen-Menü). Achtung: in diesem Fall wird kein Code generiert ! Die Ergebnisse der Prüfung werden im Meldungsfenster ausgegeben, das standardmäßig im unteren mittleren Teil der Benutzeroberfläche platziert ist.

Auch ohne vorausgehende Prüfung können Sie sich in die Steuerung einloggen (Stellen Sie zuvor sicher, dass die Steuerung läuft; Symbol in der Systemleiste ist bunt).

Wählen Sie den Befehl **Einloggen auf 'Application'** (Online-Menü). Eine Dialogbox wird geöffnet:

"Applikation Application existiert nicht auf der Steuerung. Soll die Applikation angelegt und geladen werden?" Wählen Sie "Ja" um Übersetzung und Download der Applikation zu starten.

Die Übersetzungsmeldungen werden im Meldungsfenster ausgegeben. Bei korrekter Eingabe des Beispiels sind keine Übersetzungsfehler zu erwarten, so dass die Applikation auf der Steuerung nun gestartet werden kann, siehe (11).

(10) Applikation starten und "monitoren"

Wenn das "Standardprojekt" wie in den vorausgegangenen Schritten beschrieben erstellt und geladen wurde, kann nun die fehlerfreie Applikation 'Application' auf dem Steuerungsgerät L-force Controller 3200 ("PLCWinNT) verwendet werden.

Starten der Applikation auf der Steuerung:

Wählen Sie Befehl **Start 'Application...'** im Online-Menü.

Das Programm läuft. Die Einträge für Steuerung und Applikation im Gerätefenster werden grün hinterlegt und hinter 'Application' steht "[run]". Ein grünes LÄUFT wird in der Statuszeile am unteren Rand der Programmiersystem-Oberfläche angezeigt.

Überwachen ("Monitoren") der Applikation:

Es gibt zwei Möglichkeiten, die Variablen des Applikationsprogramms zu überwachen bzw. über die Überwachungsfenster auch zu beeinflussen:

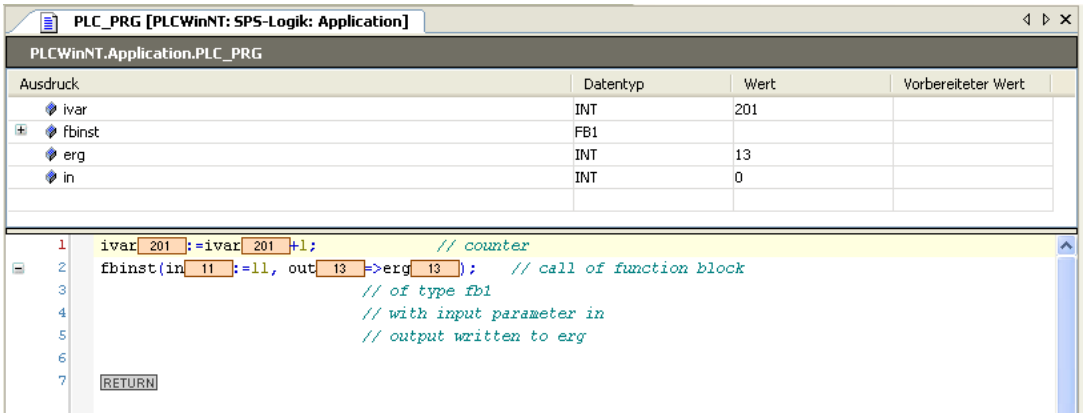
1. Online-Ansichten der einzelnen Bausteine
2. Schreiben und Forcen von Variablenwerten
3. Definierte Variablenlisten in separaten "Überwachen"-Ansichten

1. Online-Ansicht eines Programmbausteins

Diese Ansicht einer POU zeigt die aktuellen Werte der in ihr enthaltenen überwachbaren Ausdrücke in einer Tabelle im Deklarationsteil und - wenn aktiviert - als "Inline-Monitoring" auch im Implementationsteil.

Um die Online-Ansicht zu öffnen, führen Sie im POUs-Fenster einen Doppelklick auf "PLC_PRG" aus oder wählen Sie diesen Eintrag aus und verwenden den Befehl 'Objekt bearbeiten' aus dem Kontextmenü.

Im unteren Teil der Ansicht sehen Sie die Code-Zeilen wie im Offline-Modus eingegeben, ergänzt durch die kleinen Fenster des "Inline-Monitorings" hinter jeder Variablen, die deren aktuellen Wert zeigen. Im oberen Teil sind die überwachbaren Ausdrücke des Bausteins in einer Tabelle dargestellt, d. h. die entsprechenden Variablen mit Datentyp und aktuellem Wert.



2. Schreiben und Forcen von Variablen:

Sie können einen "Vorbereiteten Wert" beispielsweise für Variable ivar auf die Steuerung schreiben oder forcen, was heißt, dass ivar diesen Wert auf der Steuerung zu Beginn des nächsten Abarbeitunszyklus erhält (und bei "Forcen" behält). Wählen Sie zu diesem Zweck das Feld in der Spalte **Vorbereiteter Wert** aus, öffnen mit der Leertaste ein Eingabefeld, geben einen Integer-Wert ein und schließen mit der Eingabetaste oder einem Mausklick außerhalb des Feldes. Führen Sie dann den Befehl "Werte schreiben" bzw. "Werte forcen" im Online-Menü aus. Sie sehen das entsprechende Ergebnis in Spalte "Wert".

3. Überwachungs-Listen verwenden:

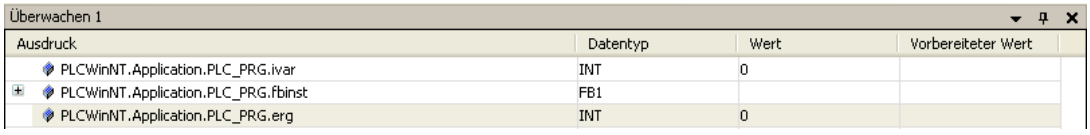
So genannte "Überwachen"-Ansichten können verwendet werden, um darin zu überwachende Ausdrücke aus der Applikation in verschiedenen Listen zusammenzustellen. Dies kann zu "Debugging-Zwecken" nützlich sein, wenn bestimmte Variablen auf einen Blick überwacht werden können sollen.

Wählen Sie den Befehl "Überwachen - Überwachen 1" aus dem Menü **Ansicht**. Das Überwachungsfenster wird geöffnet.

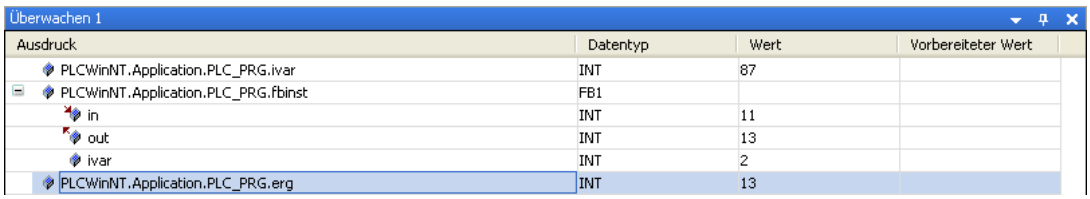
Klicken Sie in Spalte **Ausdruck** in die erste Zeile um ein Eingabefeld zu erhalten. Geben Sie den kompletten Pfad für die zu überwachende Variable ein, am besten verwenden Sie dazu die Eingabehilfe über die Schaltfläche ...:

"PLCWinNT.Application.PLC_PRG.ivar". Schließen Sie mit der Eingabetaste. Der Datentyp der Variablen wird automatisch in der Zeile eingefügt.

Fügen Sie weitere Zeilen für die anderen Variablen ein. Die unten gezeigte Überwachungsliste enthält nur Ausdrücke aus PLC_PRG, aber natürlich kann ein beliebiges Set an Variablen aus allen Bausteinen des Projekts zusammengestellt werden. Beachten Sie für Instanz-Variablen wie z. B. für die der FB1-Instanzen, dass es genügt, den Ausdruck "PLCWinNT.Application.PLC_PRG.fbinst" einzugeben. Die einzelnen Parameter werden automatisch eingefügt und die entsprechenden Zeilen können über das Plus-Symbol vor der Zeile geöffnet werden. Der aktuelle Wert jedes Ausdrucks wird in Spalte Wert angezeigt:



Wenn noch nicht erfolgt, starten Sie die Applikation mit Befehl "Start 'Application...'" im Online-Menü. Das Programm läuft nun auf der Steuerung und die aktuellen Werte werden in Spalte **Wert** angezeigt:



Schreiben und Forcen von Werten ist hier auf die gleiche Weise möglich wie oben in 2. beschrieben.

Um die Verbindung zur Steuerung zu beenden, führen Sie den Befehl 'Ausloggen...' im Online Menü aus.

"Debuggen" einer Applikation

(11) Haltepunkt setzen, Programm schrittweise abarbeiten

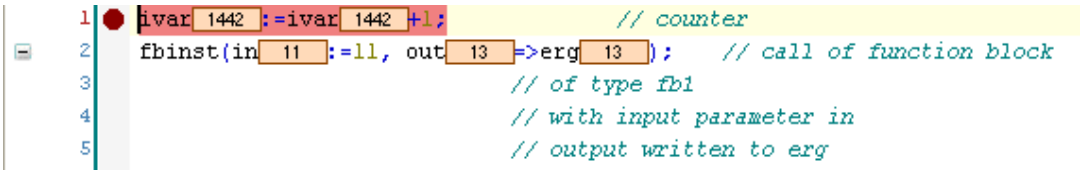
Im Online-Modus können Haltepunkte (Breakpoints) gesetzt werden, an denen die Abarbeitung des Programms jeweils gestoppt werden soll.

Wenn ein Haltepunkt erreicht wird, kann das Programm auch schrittweise abgearbeitet werden. An jedem Haltepunkt bzw. bei jedem Schritt kann der aktuelle Wert der Variablen in den Monitoring-Ansichten geprüft werden.

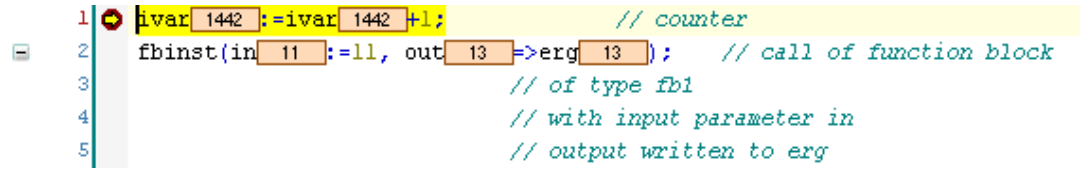
Versuchen Sie dazu folgendes:

Setzen Sie den Cursor in Zeile 1 in PLC_PRG. Drücken Sie <F9>, was dem Befehl **Haltepunkt umschalten** im Online-Menü entspricht.

Der Haltepunkt wird im Programm angezeigt; ist die Applikation gerade im Stop-Status, so sieht dies folgendermaßen aus:



Wenn die Applikation gerade läuft, stoppt sie nun am Haltepunkt:



Nun kann mit wiederholtem Drücken von <F8>, was dem Befehl **Einzelschritt** im Online-Menü entspricht, schrittweise abgearbeitet werden. Dabei wird auch in die Funktionsblock-Instanz hinein "gesteppt". Um diesen Funktionsblock-Durchlauf auszulassen, kann anstelle von <F8> mit <F10> gearbeitet werden, was dem Befehl **Prozedurschritt** entspricht. Die aktuellen Variablenwerte werden jeweils an der gerade erreichten Abarbeitungsposition angezeigt.

Sehen Sie sich auch den Haltepunkte-Dialog an, der mit Befehl "Haltepunkte" im Menü "Ansicht" geöffnet werden kann. Die aktuell definierten Haltepunkte sind hier aufgelistet und können bearbeitet werden, bzw. es können neue hinzugefügt werden.

Beachten Sie, dass die Haltepunkt-Positionen gespeichert werden, auch wenn aus der Steuerung ausgeloggt wird. Beim nächsten Einloggen werden sie als schwachrote Markierungen angezeigt und können reaktiviert werden.

5 Benutzeroberfläche

5.1 Komponenten der Benutzeroberfläche

Die Benutzeroberfläche des PLC Designer-Programmiersystems ist aus einzelnen Komponenten zusammengesetzt (Einzelheiten dazu siehe weiter unten).

Das Aussehen der Oberfläche hängt zum einen von den spezifischen **Anpassungseinstellungen** ab, die in einer Datei gespeichert sind und die Gestaltung der Menüs, Werkzeugleisten (Symbolleisten), Tastaturbedienung definieren. Die aktuellen Einstellungen können im **Anpassungen**-Dialog bearbeitet werden, der standardmäßig über das **Tools** -Menü geöffnet werden kann.

Zum anderen wird das Aussehen der Oberfläche durch die **Anordnung** der "Ansichten" und Fenster bestimmt, die der Anwender jederzeit durch Verschieben, "Andocken", "Verankern", Vergrößern/Verkleinern oder Schließen von Fenstern verändert werden kann.

Die Benutzeroberfläche stellt **Menüs** und **Symbolleisten** bereit, **Fenster** für Editor-Ansichten, Objektorganisation, Überwachung und Meldungsausgabe, sowie eine **Informations- und Statuszeile**.

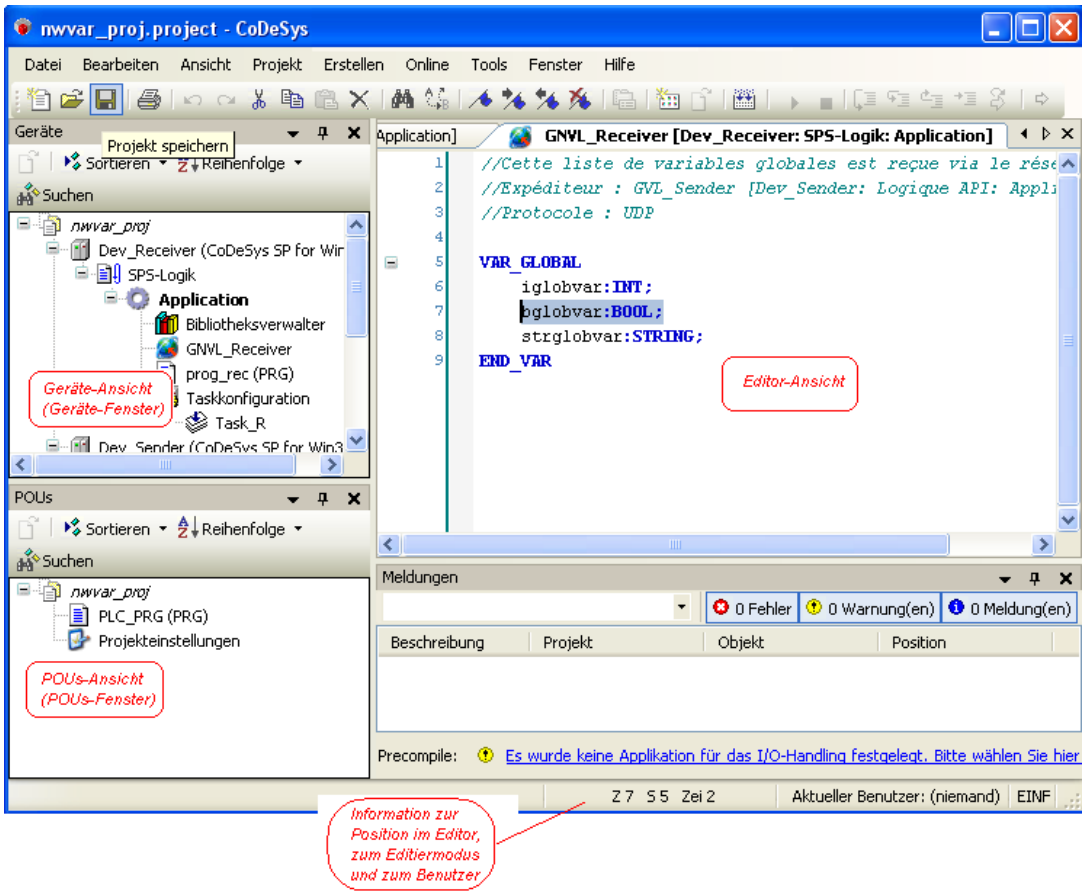


Abb. 39 Beispiel der PLC Designer V3.x Benutzeroberfläche

Die Standardkomponenten

- **Menüleiste**: Zeigt die Menüs, die gemäß der im Anpassen-Dialog vorliegenden Konfiguration die aktuell verfügbaren Kommandos enthalten. Standardmäßig sichtbar.
- **Symbolleiste**: Enthält die mit Symbolen gekennzeichneten Schaltflächen für alle Werkzeuge (Tools), die gemäß der im Anpassen-Dialog vorliegenden Konfiguration verfügbar sind plus ev. solche, die durch ein Toolbox-Plug-In eingebracht werden. Standardmäßig sichtbar.
- **ToolBox Fenster: Enthält werkzeugspezifische Elemente; kann standardmäßig über das 'Ansicht'-Menü geöffnet werden.**
- **POUs**-Fenster: Hier werden die Programmierobjekte (POUs, DUTs etc.) des Projekts in einer Baumstruktur organisiert; **kann standardmäßig über das 'Ansicht'-Menü geöffnet werden.**
- **Geräte**-Fenster: Hier werden die Geräte-Ressourcenobjekte des Projekts in einer Baumstruktur organisiert; **kann standardmäßig über das 'Ansicht'-Menü geöffnet werden.**
- **Editor**-Fenster: Ein Editor-Fenster öffnet, wenn ein Editor aktiviert wird, um ein Objekt zu bearbeiten. Bei Spracheditoren (z. B. ST-Editor, CFC-Editor) enthält das Editor-Fenster üblicherweise im unteren Teil den Spracheditor und im oberen Teil den Deklarationseditor. Bei anderen Editoren kann es auch Dialoge enthalten (z.B. Task-Editor, Geräte-Editor). Der Name der gerade bearbeiteten POU oder Geräte-Ressource wird immer in der Titelzeile des Fensters angegeben. Ein Editor-Fenster kann im Offline- wie auch im Online-Modus geöffnet werden (Doppelklick auf Objekteintrag im POUs- oder Gerätebaum bzw. über Befehl Objekt bearbeiten.

Information über die aktuellen Vorgänge im Projekt im Offline- oder Online-Modus bieten die folgenden Komponenten:

- **Meldungsfenster**: Hier werden die aktuellen Meldungen bezüglich eines Übersetzungslaufs oder eines Projekt-Downloads etc. ausgegeben.
- **Überwachungs**-Fenster und Online-Ansichten von Editoren: Erlauben das "Monitoring" einer POU bzw. einer benutzerdefinierten Liste von Ausdrücken/Variablen.

- **Informations- und Statuszeile: Die Zeile am unteren Rand der Benutzeroberfläche zeigt an, welcher Benutzer aktuell am Projekt arbeitet. Wenn gerade ein Editor-Fenster aktiv ist, wird die aktuelle Position des Cursors und der eingestellte Editier-Modus angezeigt. Im Online-Betrieb sehen Sie den augenblicklichen Status des Programms.**

  Aktueller Benutzer: Jedes Projekt hat eine Benutzer- und Rechteverwaltung. Der aktuelle Benutzer wird in der Statuszeile angegeben.

  Cursor-Position: Ausgehend vom linken bzw. oberen Rand des Editor-Fensters:

  Z = Zeilennummer

  S = Spaltennummer (eine Spalte enthält jeweils genau 1 Leerstelle bzw. Buchstaben oder Ziffer)

  Zei = Anzahl Zeichen (hier kann ein "Zeichen" ein einzelner Buchstabe, eine einzelne Ziffer oder auch eine Tabulatorspanne, die z.B. 4 Spalten umfaßt, sein)

  Nach einem Doppelklick auf eines der Felder öffnet sich der Dialog **Gehe zu Zeile**, wo eine andere Zeile angegeben werden kann, in der der Cursor platziert werden soll.

  Aktuell eingestellter Editiermodus: **EINF** = Einfügemodus, **ÜB** = Überschreibmodus. Mit einem Doppelklick auf das Feld kann die Einstellung umgeschaltet werden.
  Online-Modus Information: Status der Applikation auf der Steuerung:
  LÄUFT = Programm läuft

  STOP = Programm gestoppt

  HALT AUF BP = program halted on a breakpoin

  Programm geladen = Programm auf Steuerung geladen.

  Programm unverändert = Programm auf der Steuerung entspricht dem im Programmiersystem

  Programm geändert (Online Change) = Programm auf der Steuerung unterscheidet sich von dem im Programmiersystem, Online Change nötig

  Programm geändert (Full download) = Programm auf der Steuerung unterscheidet sich von dem im Programmiersystem, voller Download nötig

Fenster, Ansicht (View), Editor-Fenster

Die Fenster, die innerhalb des Hauptfensters der Benutzeroberfläche oder zwar dazugehörig aber daneben angezeigt werden, sehen auf den ersten Blick alle gleich aus. Man unterscheidet jedoch zwei Typen:

1. Die einen können am Rahmen des Hauptfensters **angedockt** werden oder **losgelöst** davon an beliebiger Stelle auf dem Bildschirm positioniert werden . Sie können auch "**ausgeblendet**" werden und werden dann nur durch einen Reiter im Rahmen des Hauptfensters repräsentiert. Der Inhalt solcher Fenster hängt nicht von einem bestimmten Projekt-Objekt ab, z.B. Meldungen, Geräte, POUs, Toolbox. Sie können über das '**Ansicht**'-Menü geöffnet werden und werden auch "Ansicht" ("View") genannt. Die meisten "Ansichten" enthalten eine nicht-konfigurierbare Symbolleiste mit Schaltflächen für Sortieren, Anzeigen und Suchen innerhalb der Ansicht.

2. Ein anderer Fenstertyp öffnet sich, wenn sie ein bestimmtes Objekt des Projekts im entsprechenden **Editor** ansehen oder **bearbeiten**. Diese Fenster werden alle in einem bestimmten Bereich der Benutzeroberfläche als MDI-Fenster in Form von Registerblättern angeordnet. Wie die Anordnung vorgenommen wird, hängt von den benutzerspezifischen Einstellungen ab. Sie **können nicht ausgeblendet** und **nicht vom Hauptrahmen gelöst** werden. Solche Fenster können über die das **Fenster**-Menü geöffnet werden.

Möglicherweise sind weitere Fenster- oder Ansicht-Typen aufgrund von herstellerspezifischen Komponenten vorhanden.

5.2 Benutzeroberfläche anpassen

Das aktuelle Aussehen der Benutzeroberfläche, d.h. die Anordnung und Konfiguration der einzelnen Komponenten, hängt von Folgendem ab:

- Einstellungen für Menüs, Tastaturfunktionen und Symbolleisten. Mit dem PLC Designer werden bestimmte Standardeinstellungen installiert, die jedoch lokal vom Anwender in den **Anpassen**-Dialogen (typischerweise im Menü 'Tools') überschrieben werden können. Jeweils die aktuellen Einstellungen werden dann lokal gespeichert und verwendet. Über eine Reset-Funktion können jederzeit wieder die Standardwerte gesetzt werden.
- Eigenschaften von Editoren, wie im entsprechenden Optionen-Dialog gesetzt. Auch hier können die Standardeinstellungen vom Anwender überschrieben werden und die aktuelle Konfiguration wird ebenfalls lokal gespeichert.
- Anordnung von Fenstern innerhalb des Projekts durch den Anwender. Diese wird mit dem Projekt gespeichert (siehe unten).

Anpassen der Menü- und Symbolleisten

Die Menüleiste befindet sich immer am oberen Rand des Hauptfensters der Benutzeroberfläche, zwischen Titelleiste und Fensterbereich. Eine Symbolleiste kann ebenfalls dort positioniert sein (fest) oder als unabhängiges Fenster irgendwo auf dem Bildschirm.

Um eine Leiste zu verschieben, klicken Sie mit dem Mauszeiger auf die **gepunktete Linie** am linken Ende der Leiste, halten Sie die Maustaste gedrückt und schieben die Leiste an die gewünschte Position.

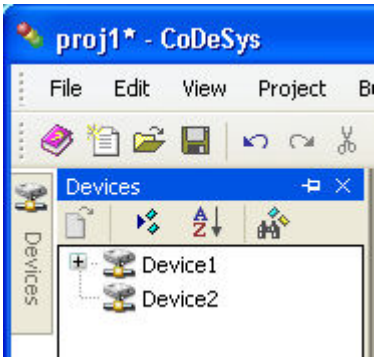


Abb. 40 Dotted line on left side of menu and tool bars

Hinweis: In Ansichten wie z.B. POUs- oder Geräte-Fenster gibt es eine **spezielle** Menüleiste mit Schaltflächen für das Sortieren, Anzeigen und Suchen im Fenster. Diese Menüleiste kann nicht konfiguriert werden.

Anordnen von Fenstern

Schließen eines Fensters: Klicken Sie auf die ⊠Schaltfläche in der oberen linken Ecke des Fensters.

Öffnen eines geschlossenen Fensters: Standardmäßig können die "Ansichten" der Standardkomponenten über den Befehl <Name> im 'Ansicht'-Menü geöffnet werden. Ein Editor-Fenster kann mit dem Befehl 'Objekt bearbeiten' oder durch einen Doppelklick auf den entsprechenden Objekteintrag im POUs- oder Geräte-Fenster geöffnet werden.

Größenänderung von Ansichten und Fenstern innerhalb des Hauptfensters: Verschieben Sie die Trennlinien zwischen benachbarten Ansichten. Losgelöste Fenster können durch Verschieben der Rahmenlinien in der Größe verändert werden.

Verschieben einer Ansicht auf dem Bildschirm bzw. innerhalb des Hauptfensters: Klicken Sie in die Titelleiste oder im Falle von Registerblättern auf den Reiter der Ansicht, halten Sie die Maustaste gedrückt und ziehen Sie die Ansicht in die gewünschte Richtung. Pfeilsymbole zeigen daraufhin die möglichen Zielpositionen an. Halten Sie die Maustaste weiterhin gedrückt und bewegen Sie den Mauszeiger auf das entsprechende Pfeilsymbol. Daraufhin wird die Zielposition als blauschattierter Bereich angezeigt.

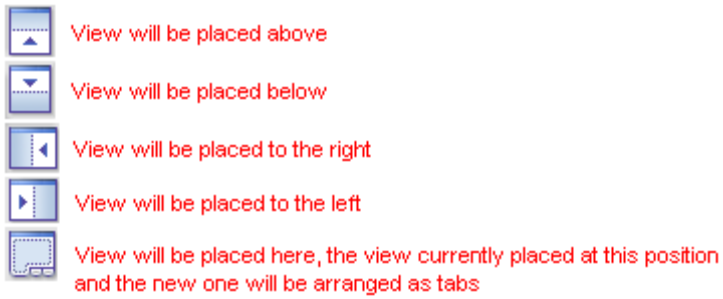


Abb. 41 Symbols indicating new position

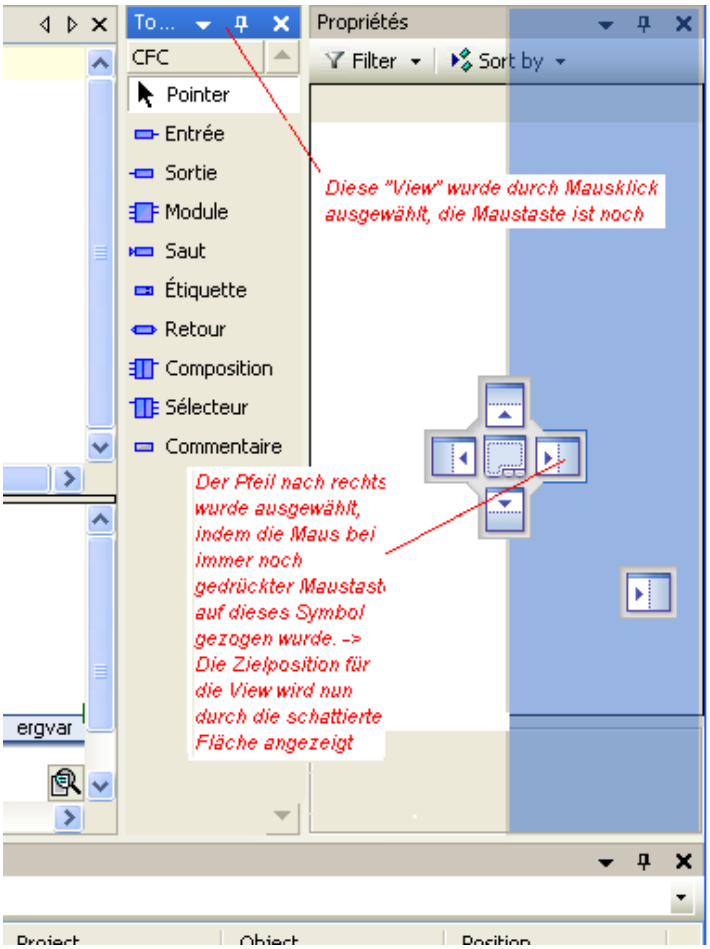


Abb. 42 Beispiel der Navigation mit Hilfe der Pfeilsymbole

Bei Loslassen der Maustaste wird die Ansicht im gewählten Zielbereich eingefügt. Ansichten mit einer **"docking"-Schaltfläche** können auch als **losgelöste** Fenster irgendwo auf dem Bildschirm platziert werden anstelle sie auf eines der oben beschriebenen Pfeilsymbole zu ziehen (verankern). In diesem Fall wird die docking-Schaltfläche "entriegelt" dargestellt. Alternativ können die Befehle 'Andocken' und 'Unverankert' verwendet werden, die standardmäßig im Fenster-Menü enthalten sind.

Ausblenden von Ansichten: Ansichten mit einer docking-Schaltfläche können ausgeblendet werden. Klicken Sie dazu auf die Schaltfläche in der oberen linken Ecke der Ansicht. Die Ansicht wird daraufhin nur mehr als ein Rechteck, das Titel und Symbol der Ansicht enthält, im Rahmen des Hauptfensters repräsentiert. Der Inhalt der Ansicht wird dann nur angezeigt, solange der Mauszeiger auf diesem Rechteck steht. Die docking-Schaltfläche verändert sich in diesem Status zu .

Um die Ansicht wieder einzublenden, klicken Sie auf die Schaltfläche. Das Aus- und Einblenden von Ansichten kann auch über die Option Ausblenden durchgeführt werden, die standardmäßig im Fenster-Menü enthalten ist.

Beachten Sie, dass die Position der **Informations- und Status-Leiste**, standardmäßig am unteren Rand des Hauptfensters, nicht verändert werden kann.

Zoom

Jedes Editor-Fenster stellt eine Zoom-Funktion bereit, um den Inhalt des Fensters zu vergrößern bzw. zu verkleinern. Die Zoom-Schaltfläche in der unteren linken Ecke des Fensters öffnet eine Auswahlliste der Zoom-Stufen. 25, 50, 75, 100, 150, 200, 400 Prozent. Beachten Sie, dass beim Ausdrucken des Fensterinhalts immer die 100%-Ansicht verwendet wird.

Die Anpassung der Benutzeroberfläche ist im **Offline- und Online-Modus möglich**.

5.3 Benutzeroberfläche im Online-Modus

Sobald mit einem Projekt auf das Zielsystem eingeloggt wird, werden alle Objekte, die im Offline-Modus in Fenstern und Ansichten geöffnet waren, im Online-Modus dargestellt.

Um ein Objekt im Online-Modus zu öffnen, das im Offline-Modus noch nicht geöffnet war, führen Sie einen Doppelklick auf den Objekteintrag im POUs- oder Gerätefenster aus oder verwenden Sie den Befehl 'Objekt bearbeiten'.

Ist Ihre Auswahl eindeutig, so wird das Objekt (Funktionsblock etc.) im Online-Modus geöffnet. Andernfalls, wenn nämlich mehrere Instanzen des ausgewählten Objekts im Projekt existieren, wird ein Dialog mit dem Titel '**Online Status auswählen <Objektname>**' erscheinen, in welchem Sie wählen können , ob eine Instanz oder aber die grundlegende Implementation des Objekts selbst geöffnet werden soll (und dies wahlweise im Online- oder Offline-Modus).

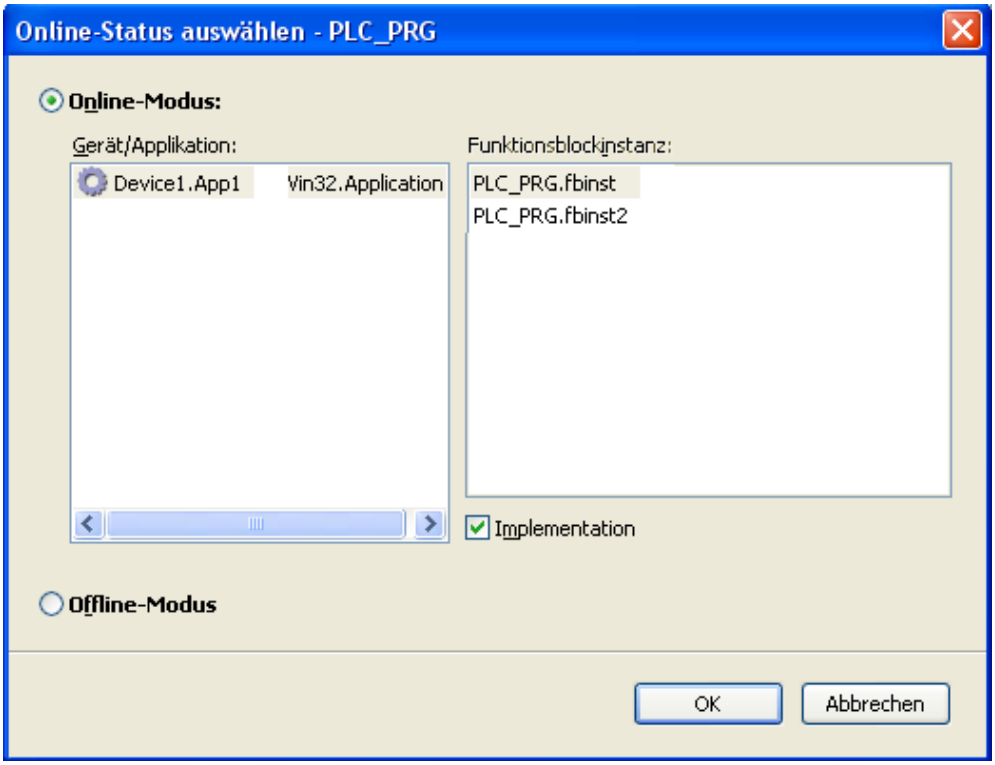


Abb. 43 Dialog 'Online-Status auswählen'

Im Feld **Gerät/Applikation** sehen Sie das Gerät und die Applikation, denen das Objekt zugeordnet ist..

Aktivieren Sie eine der Optionen **Online-Modus** oder **Offline-Modus** und klicken Sie OK, um die gewünschte Ansicht zu bekommen.

Wenn das Objekt ein Funktionsblock ist, erscheint im Feld **Funktionsblockinstanz** eine Liste aller momentan in der Applikation verwendeten Instanzen. In diesem Fall können Sie

- entweder eine der Instanzen auswählen und Online- oder Offline-Modus auswählen, oder
- Option **Implementation** aktivieren, die - unabhängig von der gewählten Instanz - die Basisimplementation des Funktionsblocks anzeigen wird. Diese Option hat keine Funktion für nicht-instanzierte Objekte.

Für weitere Informationen zu den Online-Ansichten der einzelnen Editoren sehen Sie bitte die jeweilige Editor-Beschreibung.

Die Statuszeile enthält Information zum aktuellen Status der Applikation.

6 Standard Menüs und Befehle

Sehen Sie im folgenden eine Übersicht über die Standardstruktur der Hauptmenüs und Befehle, die in einem "Standardprojekt" vorliegt.

Weitere Befehle sind verfügbar und können in bereits definierten oder neu definierten Menüs eingefügt werden. Editorspezifische Menüs und Befehle erscheinen nur, wenn der betreffende Editor geöffnet ist (Beispiel: Bibliotheksverwalter und "Bibliotheken", AS-Editor und Menü "AS").

Die Menüstruktur und die Tastaturbelegungen können jederzeit im Dialog Anpassen (Tools Menü) geändert werden.

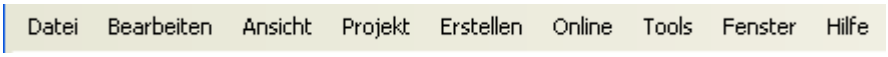


Abb. 44 Standard Menüleiste

Standardmenüs:

Datei: Befehle für Aktionen zur Projektdatei (Öffnen, Schließen, Speichern, Drucken, Seiteneinstellungen, Projekt auf/von Steuerung laden...).

	Neues Projekt...	<Strg>+<N>
	Projekt öffnen...	<Strg>+<O>
	Projekt schließen	
	Projekt speichern	<Strg +<5>
	Projekt speichern als...	
	Projekt speichern und ins Bibliotheks-Repository installieren	
	Projektarchiv	
	Extract Archive...	
	Save/Send Archive	
	Quellcode-Download	
	Quellcode aus Steuerung laden...	
	Drucken	
	Seiteneinstellungen...	
	Zuletzt verwendete Projekte >	
	<n> <Projektpfad>	
	...	
	Beenden	<Alt>+<F4>

Bearbeiten: Befehle zum Arbeiten in Texteditoren (Spracheditoren, Deklarationseditor...).

	Rückgängig	<Strg>+<Z>
	Wiederherstellen	<Strg>+<Y>
	Ausschneiden	<Umsch>+<Entf>
	Kopieren	<Strg>+<Ins>
	Einfügen	<Umsch>+<Ins>
	Löschen	<Entf>
	Alles selektieren	<Strg>+<A>
	Suchen & Ersetzen >	
	Suchen	<Strg>+<F>
	Ersetzen	<Strg>+<H>
	Weitersuchen	<F3>
	Weitersuchen (Auswahl)	<Strg>+<F3>
	Vorheriges suchen	<Umsch>+<F3>
	Vorheriges suchen (Auswahl)	<Umsch>+<Strg>+ <F3>
	Datei als Text einfügen	
	Browse	
	Gehe zur Definition	
	Erweitert >	
	Überschreibmodus	<Einf>
	Gehe zu Zeile...	
	In Großbuchstabenumwandeln	<Umsch>+<Strg>+C
	In Kleinbuchstaben umwandeln	<Strg>+<V>
	Gehe zur zugehörigen Klammer	
	Bis zur zugehörigen Klammer selektieren	
	Lesezeichen >	
	Lesezeichen ein-/ausschalten	<Strg>+<F12>
	Nächstes Lesezeichen	<F12>
	Vorheriges Lesezeichen	<Umschalt>+<F12>
	Lesezeichen löschen	
	Eingabehilfe...	<F2>
	Variablen deklarieren...	<Umschalt>+<F2>
	Nächste Meldung	<F4>
	Vorherige Meldung	<Umschalt>+<F4>
	Gehe zu Quelltextposition	

Ansicht: Befehle zum Aktivieren der einzelnen Standard-Ansichten in Fenstern der Benutzeroberfläche. Beachten Sie hierzu auch das Menü 'Fenster' (s.u.).

	POUs	
	Geräte	
	Meldungen	
	Elementeigenschaften - für AS-Eelement - für Visualisierungselement	
	Werkzeuge	
	Überwachen >	
	Überwachen <n>	
	...	
	Haltepunkte	
	Querverweisliste	
	Aufrufliste	
	Startseite	
	Anstehende Eincheckvorgänge	
	Ganzer Bildschirm	<Umsch> + <Strg> + <F12>
	Eigenschaften...	

Projekt: Befehle zur Handhabung der Projektobjekte und der Projektinformationen, für das Kopieren und Exportieren von Projekten bzw. Objekten, für Quellcodeverwaltung und Benutzerverwaltung.

	Objekt hinzufügen...	
	Gerät anhängen...	
	Gerät einfügen...	
	Gerät setzen...	
	Scan for Devices...	
	Gerät aktualisieren	
	Ordner hinzufügen...	
	Objekt bearbeiten	
	Objekt bearbeiten mit...	
	Aktive Applikation setzen	
	Projektinformationen	
	Projekteinstellungen	
	Projektumgebung	
	Dokumentieren...	
	Vergleichen...	
	Exportieren	
	Importieren	
	Projekt kopieren...	
	Benutzerverwaltung	
	Benutzer anmelden...	
	Benutzer abmelden	
	Rechte...	
	Quellcodeverwaltung	
	Projekt aus Quellcodeverwaltung öffnen	
	Quellcodeverwaltung ändern	
	Objekte aus Quellcodeverwaltung hinzufügen	
	Letzte Version abrufen (Gesamtes Projekt)	
	Letze Version abrufen (Selektierte Objekte)	
	Auschecken...	
	Einchecken...	
	Auschecken rückgängi	
	Unterschiede anzeigen	
	Versionsgeschichte...	

	Projektversionsgeschichte...	
	Bezeichnung...	
	Status aktualisieren	
	Anmelden...	

Erstellen: Befehle für das Erstellen - durch Übersetzen - des Projekts (ohne Code-Erzeugung), d. h. für das Ausführen eines Precompile-Laufs mit Syntaxprüfung; außerdem Befehle für das Löschen der letzten Übersetzungsinformationen (Bereinigen), was für Online Change von Bedeutung ist sowie Offline-Code-Erzeugung.

	Übersetzen	<F11>
	Neu übersetzen	
	Code erzeugen	
	Erzeuge Laufzeitsystemdateien...	
	Bereinigen	

Online: Befehle zum Ein- und Ausloggen auf/aus der Steuerung, für das Laden des Projekts auf die Steuerung und für Reset.

	Einloggen	<Alt>+<F8>
	Ausloggen	<Strg>+<F8>
	Bootapplikation erzeugen	
	Laden	
	Online Change	
	Quellcode auf verbundene Steuerung schreiben...	
	Mehrfacher Download	
	Reset warm	
	Reset kalt	
	Reset Ursprung	

Debug: Befehle für das Kontrollieren des Programmlaufs auf der Steuerung (Start, Stop) und für Debugging-Aktionen (Haltepunkte, schrittweise Abarbeitung, Werte schreiben und forcen).

	Start	<F5>
	Stop	<Umsch>+<F8>
	Einzelzyklus	
	Neuer Haltepunkt...	
	Haltepunkt umschalten	<F9>
	Prozedurschritt	<F10>
	Einzelschritt	<F8>
	Ausführen bis Rücksprung	<Umsch>+<F10>
	Ausführen bis Cursor	
	Nächste Anweisung festlegen	
	Nächste Anweisung anzeigen	
	Werte schreiben	<Strg>+<F7>
	Werte forcen	<F7>
	Forcen für alle Werte aufheben	<Alt>+<F7>
	Darstellung >	
	Binär	
	Dezimal	
	Hexadezimal	

Tools: Befehle zum Öffnen von Werkzeugen (Tools), mit denen eine bestimmte Umgebung für das Arbeiten am Projekt geschaffen wird (Installation von Bibliotheken und Geräten, Anpassen der Benutzeroberfläche, Optionen für Editoren, Laden & Speichern etc.)

	Bibliotheks-Repository	
	Geräte-Repository...	
	Visualisierungelemente-Repository...	
	Anpassen...	
	Optionen...	

Fenster: Befehle zum Arrangieren der Fenster in der Benutzeroberfläche (Anordnung, Öffnen, Schließen etc.). Sehen Sie hierzu auch das Menü 'Ansicht', s.o..

	Nächster Editor	<Strg>+<F6>
	Vorheriger Editor	<Umsch>+ <Strg>+<F6>
	Editor schließen	<Strg>+<F4>
	Alle Editoren schließen	
	Neue horizontale Registerkartengruppe	
	Neue vertikale Registerkartengruppe	
	Unverankert	
	Andocken	
	Ausblenden	
	Nächstes Teilfenster	<F6>
	Vorheriges Teilfenster	<Umsch>+<F6>
Editor symbol	<n> <Fenstertitel> (Applikationspfad)]	
	Fenster...	

Hilfe: Befehle zum Öffnen der Online Hilfe und Informationen zum Programmiersystem.

	Inhalt...	<Umsch>+ <Strg>+<F1>
	Index...	<Umsch>+ <Strg>+<F2>
	Suchen...	
	Automation Alliance Homepage...	
	Enter The Next Level...	
	Informationen...	

7 Benutzerverwaltung und Zugriffskontrolle

Die Plug-Ins "User Management" und "User Management UI" stellen Funktionen zum Definieren von Benutzerkonten und Konfigurieren der Zugriffsrechte in einem Projekt bereit. Beachten Sie, dass ggf. eine gerätespezifische Benutzerverwaltung unterstützt wird, die die Zugriffsrechte auf das Dateisystem und Objekte der Steuerung durch den Benutzer während der Laufzeit regelt.

Das Recht, mit einer bestimmten Aktion auf Projektobjekte zugreifen zu können, wird immer nur an Gruppen vergeben, nicht an einzelne Benutzerkonten. Somit muss jeder Benutzer, der an einem Projekt arbeiten will, Mitglied einer Gruppe sein.

Benutzerverwaltung

Vor dem Anlegen von Benutzerkonten und Benutzergruppen beachten Sie bitte folgendes:

- Die Benutzer- und Gruppenkonfiguration wird im Dialog **Projekteinstellungen** vorgenommen.
- Automatisch gibt es immer eine **Gruppe "Everyone"** und standardmäßig sind zunächst jeder Benutzer und jede andere Gruppe Mitglied dieser Gruppe. Somit ist sichergestellt, dass jedes Benutzerkonto automatisch wenigstens mit den definierten Standard-Rechten ausgestattet ist. Gruppe "Everyone" kann nicht gelöscht werden, nur umbenannt, und es können keine Mitglieder aus dieser Gruppe entfernt werden.
- Ebenso gibt es automatisch immer eine **Gruppe "Owner"**, die einen **Benutzer "Owner"** enthält. Es können dieser Gruppe weitere Benutzer hinzugefügt oder welche daraus entfernt werden, aber mindestens ein Mitglied muss enthalten bleiben. Diese Gruppe kann ebenfalls nicht gelöscht werden und besitzt immer alle Zugriffsrechte. Somit wird auf jeden Fall verhindert, dass ein Projekt durch das Verneinen aller Zugriffsrechte für alle Gruppen unbrauchbar gemacht wird. Sowohl Gruppe und Benutzer "Owner" können umbenannt werden.
- Wenn das Programmiersystem bzw. ein Projekt neu gestartet wird, ist zunächst kein Benutzer im Projekt eingeloggt. Danach allerdings kann der Anwender sich über ein bestimmtes Benutzerkonto mit Benutzername und Passwort einloggen, um einen bestimmten Satz an Zugriffsrechten zu haben.
- **Beachten Sie, dass jedes Projekt seine eigene Benutzerverwaltung hat!** Um also beispielsweise bestimmte Zugriffsrechte auf eine im Projekt eingebundene Bibliothek zu bekommen, muss sich der Anwender explizit und separat in diese Bibliothek einloggen. Benutzer und Gruppen, die in verschiedenen Projekten definiert sind, sind nicht identisch, auch wenn sie identische Namen haben.

Achtung: Die Benutzer **Passwörter werden irreversibel gespeichert**. Wenn ein Passwort vergessen wird, wird das Benutzerkonto unbrauchbar. Wenn das "Owner"-Passwort vergessen wird, wird möglicherweise das gesamte Projekt unbrauchbar!

Zugriffskontrolle

Benutzerverwaltung in einem Projekt ist nur sinnvoll, wenn sie mit entsprechender Rechtevergabe für den Zugriff auf Projekt und Objekte verbunden wird. Beachten Sie bitte folgendes:

- In einem **neuen Projekt** sind grundsätzlich alle Rechte mit einem **Standardwert** (Standardrecht) gesetzt. Dieser Standardwert ist üblicherweise: "erlaubt".
- Danach kann **jedes Recht explizit vergeben ("erlaubt") oder genommen ("verboten")** bzw. auf den Standardwert zurückgesetzt werden. Die Verwaltung der Zugriffsrechte wird im Rechte-Dialog oder - für Objektzugriffsrechte - im Zugriffskontrolle-Dialog vorgenommen, der Teil des 'Eigenschaften'-Dialogs ist.
- Zugriffsrechte auf Objekte können "**vererbt**" werden. Wenn ein Objekt ein "Vater"-Objekt (Beispiel: wenn einem Programm eine Aktion zugewiesen ist, diese also im Strukturbaum unterhalb des Programms eingehängt ist, ist das Programm-Objekt "Vater" des Aktions-Objekts) hat, werden die Rechte des Vaters automatisch als Standardwerte für das Kind verwendet. Vater-Kind Beziehungen von Objekten bezüglich der Zugriffsrechte entsprechen normalerweise den Beziehungen, die im POUs- oder Gerätebaum dargestellt sind, und werden im 'Rechte'-Dialog gemäß der Syntax "<Vaterobjekt>.<Kindobjekt>" angezeigt. Beispiel: Aktion ACT ist dem POU-Objekt PLC_PRG zugewiesen. Also wird im POUs-Fenster ACT im Objektbaum unterhalb PLC_PRG eingerückt dargestellt. Im 'Rechte'-Dialog wird ACT durch "PLC_PRG.ACT" dargestellt, was zeigt, daß PLC_PRG der "Vater" von ACT ist. Wenn das "modify"-Recht explizit für PLC_PRG und eine bestimmte Benutzergruppe "verboten" wird, wird "verboten" auch zum Standardwert des "modify"-Rechts für ACT.

8 Visualisierung

8.1 Visualisierung in PLC Designer V3

Allgemeiner Mechanismus

Im Unterschied zur Version V2.x sind die Remote-Visualisierungs-Clients nichts anderes als Interpreter von Zeichenkommandos. Unabhängig von seinem Typ (Web-Visualisierung, Target-Visualisierung, HMI) bekommt jeder Client dieselben Anweisungen, so dass für jeden Typ dieselbe Visualisierung resultiert.

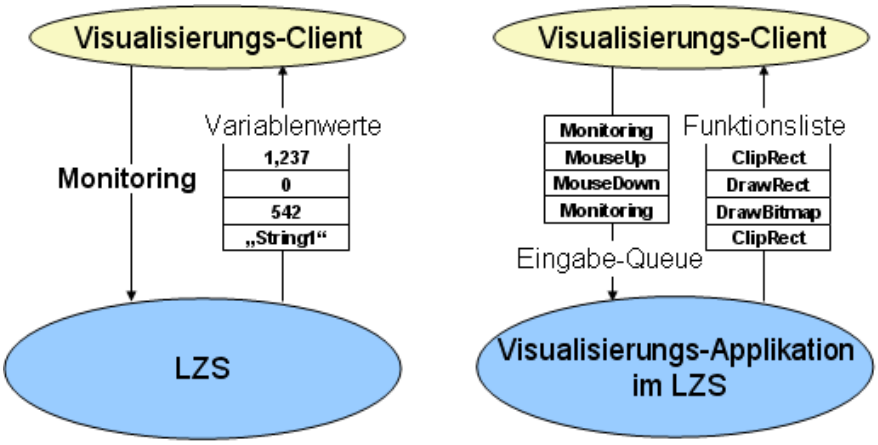


Abb. 45 Vergleich der Visualisierungs-Mechanismen - Version V2.x (links) gegenüber Version V3.x (rechts)

Als Konsequenz muss die zugehörige Visualisierungsapplikation immer auf die SPS heruntergeladen werden, was für kleine Steuerungen ein Problem darstellen kann. Um dieses Problem zu umgehen wird ein Data Server in das Laufzeitsystem integriert, welcher die von der Visualisierungs-Applikation benötigten Daten bereitstellt.

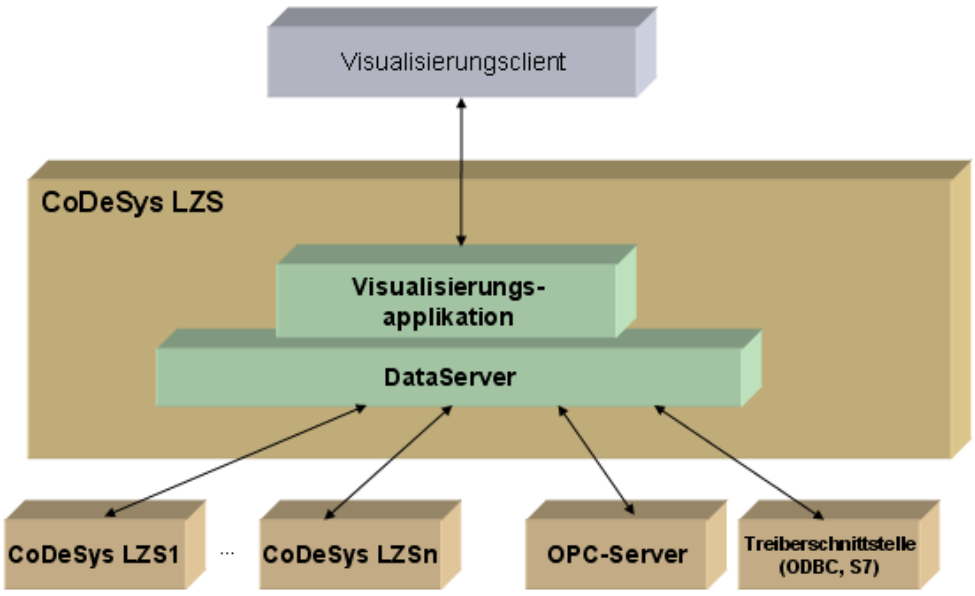
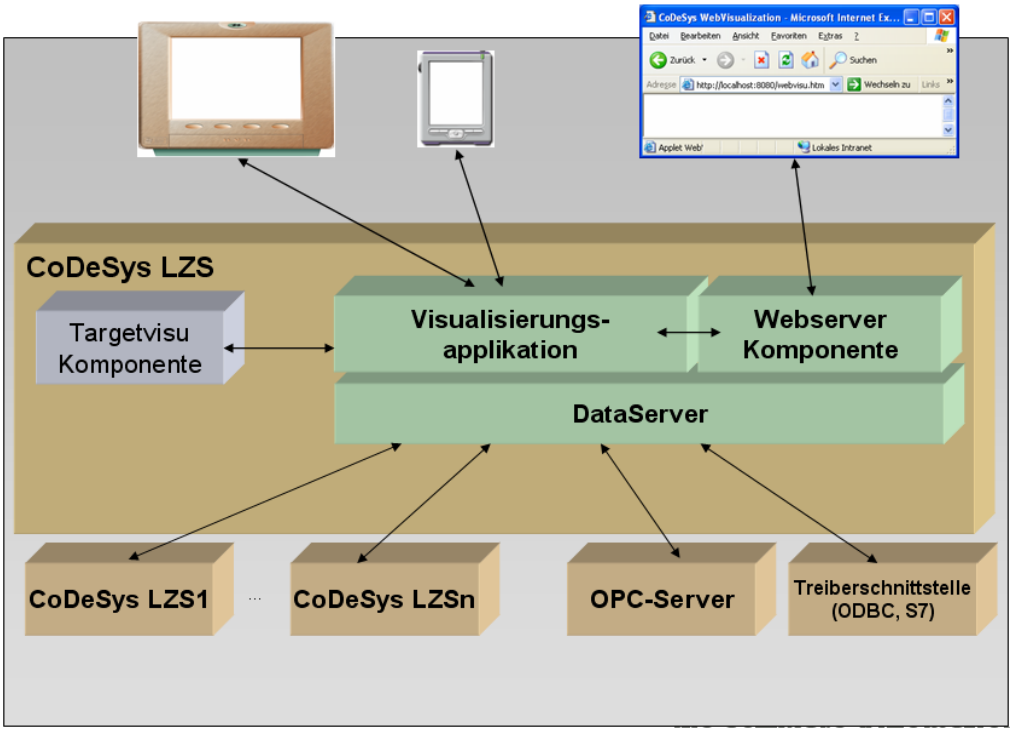


Abb. 46 Data Server für die Bereitstellung der Visualisierungsdaten

Verschiedene Visualisierungsarten

Eine Visualisierungsapplikation, d.h. die Visualisierunselemente werden in IEC-Code im Visualisierungseditor des Programmiersystems angelegt. Falls die Visualisierung auf dem Gerät laufen soll, wird Visualisierungscode erzeugt und dorthin geladen. Die möglichen vier Visualisierungsvarianten sind im folgenden beschrieben. Welche von ihnen verwendet wird, hängt davon ab, ob und welche Client-Objekte unterhalb des Visualization Manager eingefügt sind.



- Target-Visualisierung, welche Steuerung und Visualisierung auf einem Gerät kombiniert.
- PLC Designer-HMI um die Visualisierung auf einem externen Gerät darzustellen.
- Web-Visualisierung, welche zur Verbindung mit der zugehörigen Applikation einen Web-Server nutzt.
- Diagnose Visualisierung, die nur im Programmiersystem läuft, entsprechend der reinen Online-Visualisierung in PLC Designer V2.x .

Allen Visualisierungen sind folgende Eigenschaften gemein:

- Innerhalb einer Visualisierung sind beliebige Ausdrücke, sogar Funktionsaufrufe, zulässig.
- Im Gegensatz zu PLC Designer V2.3 werden alle Visualisierungselemente, wie auch die Visualisierungsobjekte selbst als IEC61131-3 Funktionsbausteine implementiert. Somit können Visualisierungen in anderen Visualisierungen instanziert, d.h. referenziert, werden und sie können verwendet werden, um andere Funktionsbausteine zu erweitern. Ein Platzhalter-Konzept ist vorhanden, das Platzhalter als Eingabe-Variablen einer Visualisierung behandelt. Sie können in einem Schnittstellen-Editor bearbeitet werden. Die Verwendung von Frame-Elementen erlaubt ein einfaches Wechseln der Anzeige von verschiedenen anderen Visualisierungen innerhalb eines Visualisierungsobjekts.
- Eine Visualisierung kann als Objekt im globalen Objekte-Pool, d. h. im POUs-Fenster angelegt werden, oder direkt einer Applikation zugeordnet im Geräte-Fenster.
- Visualisierungen im Gerätebaum werden pro Applikation durch einen Visualization Manager verwaltet, der gemeinsame Einstellungen für alle applikationsspezifischen Visualisierungen definiert.
- Einige allgemeine Einstellungen für alle Visualisierungen in einem Projekt (Dateipfade, Größenanpassung etc.) können in den Visualisierungs-Optionen vorgenommen werden.
- Jede einzelne Visualisierung hat Eigenschaften wie beispielsweise ihre geplante Verwendung (als "Visualisierung", "Numpad/Keypad" oder "Dialog") oder die Anzeigegröße. Beachten Sie in diesem Zusammenhang, dass eine Visualisierung explizit für die Verwendung als Benutzer-Eingabe-Dialog in anderen Visualisierungen angelegt werden kann. Implizit gibt es auch eine vorgefertigte Numpad- und Keypad-Maske für diesen Zweck. Die Verwendung solcher Numpad/Keypad-Vorlagen und Dialoge kann in der Konfiguration eines Visualisierungselements definiert werden.
- Der Visualisierungs-Editor wird durch eine Werkzeuge-Box (Toolbox) ergänzt, die die verfügbaren Visualisierungselemente bereitstellt, und durch einen Eigenschaften-Editor für die Konfiguration der eingefügten Elemente. Die Elemente werden durch entsprechende Bibliotheken gemäß dem aktuell eingestellten Profil geliefert. Die Visualisierungselemente können auf einfache Weise angeordnet und gruppiert werden. Die Standard-Elemente werden über entsprechende Bibliotheken im Projekt bereitgestellt. PLC Designer V2.3 Visualisierungen können importiert werden.
- Die Visualisierungselemente können über PLC Designer-Projektvariablen animiert werden, die direkt eingetragen werden, oder über Ausdrücke, d.h. Kombinationen der Variablen mit Operatoren und Konstanten. Beispielsweise erlaubt dies eine Skalierung von Variablenwerten, so dass sie für die Verwendung in der Visualisierung geeignet sind.
- Sprachumschaltung (ANSI oder UNICODE) innerhalb einer Visualisierung ist durch die Verwendung von Textlisten möglich.

- Die Variablenwerte einer Applikation können über ein Textfeld-Element in der Visualisierung verändert bzw. angezeigt werden. Die Formatierung dieser Ein- und Ausgaben basieren auf der Standardfunktion sprintf (C-Bibliothek).
- Externe Datenquellen können verwendet werden.
- Jedem Visualisierungselement kann ein Tooltip-Text zugeordnet werden.
- Neben der Zoom-Funktion ist auch automatisches Skalieren (Visualisierung paßt sich der Bildschirmgröße an) möglich.
- Visualisierungen können in Bibliotheken gespeichert und somit in anderen Projekten verfügbar gemacht werden.
- Visualisierungselemente-Repositories werden zur Verwaltung der Visualisierungselemente und -profile auf dem lokalen System verwendet.

Hinweis: Die Visualisierung arbeitet mit einem integrierten Laufzeitsystem, deshalb erscheinen während der Arbeit im Editor ggfs. Start- und Download-Meldungen.

8.2 Voraussetzungen

Ein Projekt, das eine Visualisierung enthalten soll, benötigt dieselbe Grundstruktur wie ein Projekt ohne Visuailsierung. Es muss also eine Gerätekonfiguration mit Applikation und Taskkonfiguration sowie ein Applikationsprogramm definieren.

Da die PLC Designer V3.x-Visualisierung nach Standard IEC61131-3 realisiert ist, müssen bestimmte Bibliotheken im Projekt eingebunden sein, die die erforderlichen Funktionen und Visualisierungselemente bereitstellen. Um die Auswahl der Visualisierungsbibliotheken und daraus wiederum genau die Auswahl an Elementen zu definieren, die im Visualisierungseditorjekt bereitstehen sollen, werden Visualisierungsprofile verwendet. Deshalb muss jedes Visualisierungsprojekt auf einem solchen Profil basieren.

8.3 Optionen

Im Optionen-Dialog können in Kategorie 'Visualisierung' u.a. Standardeinträge für die Projekteinstellungen gemacht werden, beispielsweise Visualisierungsverzeichnisse für Sprachdateien und Bilddateien, die für die Konfiguration von Visualisierungen bereitstehen sollen.

Für weitere Informationen siehe Projekt Optionen, Kategorie Visualisierung.

8.4 Visualisierungsobjekt anlegen

Um ein Visualisierungsobjekt im Projekt anzulegen verwendet man den Befehl Objekt hinzufügen. Eine Visualisierung kann im POUs-Fenster (globaler Objekte-Pool) oder applikations-spezifisch im Geräte-Fenster eingefügt werden, abhängig davon, ob sie projekt-global verwendet werden können soll, oder nur für eine bestimmte Applikation.

Sobald das erste Visualisierungsobjekt im Projekt eingefügt wird, werden die im gerade aktiven Visualisierungsprofil definierten Bibliotheken automatisch im Bibliotheksverwalter eingebunden.

Sobald die erste Visualisierung unterhalb einer Applikation eingehängt wird, wird automatisch auch das Visualization Manager-Objekt eingefügt.

Das neu angelegte Visualisierungsobjekt wird automatisch im Visualisierungs-Editor geöffnet.

8.5 Visualisierung bearbeiten

Der **Visualisierungs-Editor** zum Erstellen von Visualisierungen arbeitet in Kombination mit der **Werkzeuge**-Box, die die Visualisieungselemente aus den eingebundenen Elemente-Bibliotheken bereitstellt, sowie mit dem **Eigenschaften-**Editor zur Konfiguration der Visualisierungselemente. Standardmäßig werden beide mit dem Visualisierungs-Editor automatisch geöffnet und können ggfs. manuell aus dem Menü 'Ansicht' aufgerufen werden.

Um ein Visualisierungsobjekt zu öffnen, führen Sie einen Doppelklick auf das Objekt im POUs- oder Geräte-Fenster aus bzw. verwenden Sie den Befehl Objekt bearbeiten.

Für eine detaillierte Beschreibung zur Benutzung sehen Sie bitte:

- Visualisierungs-Editor
- Visualisierungs-Werkzeuge (verfügbare Visualisierungselemente)
- Visualisierungselement-Eigenschaften (Konfiguration der Elemente)

8.6 Visualization Manager

Wenn eine Visualisierung im Gerätebaum unterhalb einer Applikation eingefügt wird, wird automatisch ein Visualisierungsmanager-Objekt hinzugefügt.

Der "Visualization Manager" definiert gemeinsame Einstellungen für alle Visualisierungen der Applikation, z.B. Darstellungsgröße, Startvisualisierung, Einstellungen für Web- und Target-Visualisierung.

Siehe die weiteren Hilfeseiten zum Visualization Manger für eine Beschreibung der einzelnen Dialoge.

8.7 Startvisualisierung

Die "Startvisualisierung", d.h. das Visualisierungsobjekt, das als erstes nach dem Einloggen mit der Applikation auf der Steuerung angezeigt werden soll, muss im Geräte-Fenster unterhalb des betreffenden Applikationsobjekts eingehängt werden.

Wenn der Applikation nur eine Visualisierung zugewiesen ist, wird diese automatisch als Startvisualisierung verwendet.

Wenn der Applikation mehrere Visualisierungen zugewiesen sind, muss die Startvisualisierung explizit im Visualization Manager definiert werden. Daraufhin wird diese Option automatisch in allen anderen Visualisierungsobjekten deaktiviert.

8.8 Frames, Referenzen, Schnittstellen, Platzhalter

Das PLC Designer V2.3-Konzept von Visualisierungsreferenzen und Platzhaltern wird in PLC Designer V3 durch ein ähnliches ersetzt.

- Grundsätzlich kann eine Visualisierung in eine andere eingefügt und damit "referenziert" werden. Zu diesem Zweck wird das Frame-Element verwendet, das eine oder mehrere Referenzen auf Visualisierungen einschließen kann. Welche Visualisierungen dafür bereitstehen hängt von deren Positionen relativ zur gerade bearbeiteten im POUs- oder Gerätebaum ab.
- Jede Visualisierung kann eine Schnittstelle (zu öffnen über den Befehl Schnittstellen-Editor) bereitstellen, in der man die Eingabevariablen wie auch sonst für einen Funktionsblockbaustein definieren kann. Diese Eingabeparameter funktionieren als "Platzhalter". In einer Instanz (Referenz) der Visualisierung müssen sie dann durch Werte oder Ausdrücke für die spezielle Verwendung im lokalen Objekt ersetzt werden. Das Ersetzen muss im Eigenschaften-Fenster des Frame-Elementes, das die Visualisierungsinstanzen einbindet, vorgenommen werden. Beachten Sie dabei, dass die Eingabevariablen einer Visualisierungsinstanz, die einer bestimmten Applikation zugeordnet ist, gültigen Variablen dieser Applikation zugewiesen werden müssen.
- **Umschalten** von Visualisierungen: Wenn ein Frame-Element mehrere Visualisierungsreferenzen einschließt (Frame Auswahl), können Benutzereingaben auf ein anderes Visualisierungselement so konfiguriert werden, dass sie einen Wechsel der Anzeige dieser Referenzen im Frame bewirken. Die Eingabekonfiguration bietet zu diesem Zweck die Option 'Visualisierungswechsel'.
  Beispiel: Eine Visualisierung enthält drei Schaltflächen 1,2,3 und einen Frame, dem die Visualisierungen Visu1, Visu2 und Visu3 zugewiesen sind. Die Schaltflächen werden so konfiguriert (Eingabe, OnMouse-Aktionen, Visualisierungswechsel), dass eine Benutzereingabe auf eine Schaltfläche die jeweilige Visualisierung im Frame aufruft. Somit können - im Unterschied zu PLC Designer V2.3 - innerhalb eines Visualisierungselements verschiedene andere im Wechsel angezeigt werden, also eine Schalttafel visualisiert werden.

8.9 Text und Sprache in Visualisierungen

Ein Visualisierungselement kann mit einem Text versehen werden, indem dieser in den entsprechenden Feldern im Eigenschaften-Konfigurationsfenster eingegen wird, oder direkt ("inline") am Element im Editorfenster. Für letzteres kann am selektierten Element mit der <Leertaste> ein Eingabefeld geöffnet werden.

Textlisten werden verwendet, um solche Texte zu verwalten. Eine Textliste ist eine POU, die Text-Strings enthält, die durch die Kombination von Textlisten-Name und einer listeninternen Text ID innerhalb des Projekts eindeutig referenziert werden können. Verschiedene Sprach-Versionen (Landessprache) eines Texts können innerhalb einer Textliste definiert sein und somit ist jeder Text zusätzlich noch durch ein Sprachenkürzel gekennzeichnet. Zumindest die Standard-Sprache muss für jeden Text definiert sein.

Hinweis: Verzeichnisse, die Textlisten für die Verwendung in der Visualisierung bereitstellen, sind in den Projekteinstellungen, Kategorie Visualisierung definiert.

In den Text-Eigenschaften eines Visualisierungselements müssen der Textlisten-Name und die Text ID eingetragen werden (String-Format), um zu definieren, welcher Text im Element im Online-Modus angezeigt wird. Zusätzlich kann die Darstellung des Textes durch die Verwendung von Formatierungssequenzen beeinflußt werden. Da die Text ID auch über eine Variable angegeben werden kann, wird ein **dynamisches Umschalten von Texten** möglich.

Wenn die aktuell verwendete Textliste mehrere Landessprachenversionen eines Textes definiert, kann die in der Visualisierung zu verwendende Sprache mit Hilfe eines Visualisierungselements umgeschaltet werden (Sprachumschaltung), das entsprechend konfiguriert ist. Dazu muss es mit einer Eingabe-Eigenschaft versehen sein, die eine Mausaktion zur "Sprachumschaltung" definiert.

Somit sind eine dynamische Anzeige und Sprachumschaltung für Visualisierungstexte möglich.

Grundsätzlich gibt es zwei Typen von Text in einer Visualisierung:

1. Statische Texte:

Statischer Text kann im Online-Modus nicht verändert werden (nur die Landessprache wird, wenn in der Textliste entsprechende Sprachversionen definiert sind, immer automatisch derjenigen angepaßt, die gerade in der Visualisierung verwendet wird). Statischer Text wird in Kategorie 'Texts' in den Visualisierungs-Elementeigenschaften konfiguriert und wird in einer automatisch angelegten Textliste 'GlobalTextList' verwaltet.

2. Dynamische Texte

Dynamischer Text kann im Online Modus durch eine Benutzereingabe oder über eine Variable der Applikation verändert werden. Er wird in Kategorie 'Dynamic texts' in den Visualisierungs-Elementeigenschaften konfiguriert. Dabei muss der Name der dynamischen Textliste und die Text ID des gewünschten Texts angegeben werden.

Formatierung des Textes

Neben der Eingabe des reinen Textes in der Elementkonfiguration können auch Formatierungsangaben eingefügt werden, um die Textanzeige im Online-Betrieb zu gestalten. Eine Formatierungsangabe beginnt immer mit einem "%", gefolgt von einem Zeichen, das die Art der Formatierung bestimmt. Sie kann einzeln oder in Kombination mit dem eigentlichen Text verwendet werden.

Wenn in der Textangabe ein "**%s**" enthalten ist, wird dieses im Online-Modus durch den Wert der Variable ersetzt, die im Eigenschaften-Feld 'Text variable' in Kategorie 'Text variables' angegeben ist. Falls Sie den Instanznamen einer Variablen, die einem Visualisierungsbaustein als Eingangsparameter übergeben wurde, anzeigen lassen wollen, so ist vom dem Pragma Attribut 'parameterstringof' Gebrauch zu machen.

Beachten Sie, dass jede Formatierungsangabe verwendet werden kann, die der Standard-C-Bibliotheksfunktion 'sprintf' entspricht, wenn sie zum Datentyp der verwendeten Variable paßt.

Siehe im folgenden einige Beispiele von Formatierungsangaben und ihren "Argumenten".:

Zeichen nach "%"	Argument / Ausgabe als
d,i	Dezimalzahl
b	Binärzahl
o	Vorzeichenlose Oktalzahl (ohne vorangestellte Nulll)
x	Vorzeichenlose Hexadezimalzahl (ohne vorangestelltes "0x")
u	Vorzeichenlose Dezimalzahl
c	Einzelnes Zeichen
s	Zeichenfolge: Dieses wird im Online-Betrieb durch den Wert der Variablen ersetzt, die im Eigenschaften-Feld 'Text variable' in Kategorie 'Text variables' angegeben ist.
f	REAL-Werte; Syntax: `%|<Ausrichtung><Mindestbreite><Genauigkeit>|f` Ausrichtung wird durch Minuszeichen (linksbündig) bzw. Pluszeichen (rechtsbündig, Default) definiert; Genauigkeit definiert die Anzahl der Stellen hinter dem Komma (Default: 6).Beispiel siehe unten.

Beispiel:

Eintrag in 'Text' Eigenschaftenfeld: `Fill level %2.5f mm`

Eingabe im Feld 'Text variable' (REAL-Variable zur Angabe des Füllstandes), z.B.:
`plc_prg.fvar1`

-> Ausgabe im Online-Betrieb z.B. `Fill level 32.8999 mm`

Hinweis: Wenn ein Prozentzeichen % in Kombination mit einer der oben beschriebenen Formatierungsangaben angezeigt werden soll, muss "%%" eingegeben werden. Beispiel: Geben Sie "Rate in %%: %s" ein, um im Online-Betrieb folgendes zu erhalten: "Rate in %: 12" (wenn die Textvariable gerade "12" liefert).

Unterstützung des Unicode-Formats kann aktiviert werden. Siehe die Beschreibung zu Textlisten für genauere Informationen.

Ausgabe der Systemzeit

Eine Kombination aus"**%t**" und nachfolgenden speziellen Platzhaltern innerhalb von eckigen Klammern wird im Online-Modus durch die aktuelle Systemzeit ersetzt. Die Platzhalter definieren das Anzeigeformat, siehe untenstehende Tabelle.

Hinweis: Import von V2.3 Projekten: Die im PLC Designer V2.3 verwendete Zeitformatierung %t wird automatisch in das neue %t[] Format konvertiert, wenn ein altes Projekt importiert wird, jedoch werden die folgenden Platzhalter nicht mehr unterstützt: %U, %W, %z, %Z.

Gültige Platzhalter:

ddd	Name des Wochentags, abgekürzt, z.B. "Wed"
dddd	Name des Wochentags, z.B. "Wednesday"
ddddd	Wochentag als Zahl (0 – 6; Sonntag ist 0)
MMM	Name des Monats, abgekürzt, z.B. "Feb"
MMMM	Name des Monats, z.B. "February"
d	Tag im Monat als Zahl (1 – 31), z.B. "8"
dd	Tag im Monat als Zahl (01 – 31), z.B. "08"
M	Monat als Zahl (1 – 12), z.B. "4"
MM	Monat als Zahl (01 – 12), z.B. "04"
jjj	Tag im Jahr als Zahl (001-366), z.B. "067"
y	Jahr ohne Angabe des Jahrhunderts (0-99), z.B. "9"
yy	Jahr ohne Angabe des Jahrhunderts (00-99), z.B. "09"
yyy	Jahr mit Angabe des Jahrhunderts, z.B. "2009"
HH	Stunde, 24-Stunden-Format (01-24), z.B. "16"
hh	Stunde, 12-Stunden-Format (01-12), z.B. "8" für 16 Uhr
m	Minuten (0-59), ohne vorangestellte Null, z.B. "6"
mm	Minuten (00-59), mit vorangestellter Null, z.B. "06"
s	Sekunden (0-59), ohne vorangestellte Null, z.B. "6"
ss	Secunden (00-59), mit vorangestellter Null, z.B. "06"
ms	Milliseconds (0-999), without preceded null, for example "322"
t	Kennzeichner für die Anzeige im 12-Stunden-Format: A (Stunden <12) bzw. P (Stunden >12), z.B. "A" wenn es 9 Uhr morgens ist
tt	Kennzeichner für die Anzeige im 12-Stunden-Format: AM (Stunden <12) bzw. PM (Stunden >12), z.B. "AM" wenn es 9 Uhr morgens ist
' '	Zeichenfolgen, die einen der oben aufgeführten Platzhalter enthalten, müssen in einzelne Hochkommas gefaßt werden; alle anderen Texte innerhalb des Format-Strings können ohne Hochkommas stehen; z.B. 'update', weil ein "d" und ein "t" enthalten sind

Beispiel:

```
%t['Last update:' ddd MMM dd.MM.yy 'at' HH:mm:ss]
```

-> Anzeige im Online-Modus: `Last update: Wed Aug 28.08.02 at 16:32:45`

Font (Schriftart)

Der Font (Schriftart), der im Online-Betrieb verwendet wird, kann ebenfalls in den Visualisierungselement-Eigenschaften definiert werden. Siehe hierzu die Kategorien 'Text properties' und 'Font variables'.

Ausrichtung

Die horizontale und vertikale Ausrichtung des Elementtextes wird in den Element-Eigenschaften in Kategorie 'Text properties' definiert.

8.10 Bilder in einer Visualisierung

Ein Bild aus einer externen Bilddatei kann in einer Visualisierung mit Hilfe des Elements 'image' eingefügt werden.

Hinweis: Verzeichnisse, die Bilddateien für die Verwendung in der Visualisierung bereitstellen, sind in den Projekteinstellungen, Kategorie Visualisierung definiert.

Die Bilddateien werden im Projekt in **Bildersammlungen** verwaltet.

Die einzelnen Bilder können über eine Kombination aus dem Namen der Bildersammlung und einer ID, die innerhalb der Sammlung für jede Bilddatei vergeben ist, innerhalb des Projektes eindeutig referenziert werden.

In den Element-Eigenschaften eines Bild-Elementes können diese ID und (für eineindeutigen Zugriff) ggfs. der Name der Bildersammlung angegeben werden um festzulegen, welches Bild im Online-Betrieb angezeigt wird. Die Bildreferenz kann auch über eine Variable spezifiziert werden, was ein **dynamisches Umschalten der Bildanzeige** erlaubt.

Außerdem kann für eine Visualisierung ein Hintergrundbild definiert werden (Befehl 'Hintergrund').

8.11 Externe Datenquellen

Externe (remote) Datenquellen können auch in Visualisierungen verwendet werden. Zu diesem Zweck muss ein Data Server für die lokale Applikation angelegt sein, der die entsprechenden Datenquellen handhabt.

8.12 Visualisierungsbibliotheken

Da die Visualisierungselemente in PLC Designer V3.x als Funktionsblöcke angelegt sind, werden sie über Bibliotheken bereitgestellt.

Wenn ein Visualisierungsobjekt im Projekt eingefügt wird, werden bestimmte Visualisierungsbibliotheken im Bibliotheksverwalter eingebunden. Die Namen und Versionen dieser Bibliotheken sind im aktuell verwendeten Visualisierungsprofil definiert. Das Profil legt auch genau fest, welche Elemente aus diesen Bibliotheken in der Werkzeug-Box des Visualisierungs-Editors bereitstehen.

Eine Visualisierungsbibliothek ist immer als spezieller Typ einer "Platzhalter-Bibliothek" angelegt. Das bewirkt, dass die genaue Version der zu verwendenden Bibliothek nicht festgelegt ist, solange sie nicht in einem Projekt eingebunden wird. Erst dann bestimmt das aktuelle Visualisierungsprofil, welche Version tatsächlich benötigt wird. Beachten Sie, dass dieser Typ von Bibliothek sich von den gerätespezifischen Platzhalter-Bibliotheken unterscheidet, bei denen die Platzhalter aus der Gerätebeschreibung aufgelöst werden.

Siehe nachfolgend die Basis-Bibliotheken, die standardmäßig eingebunden werden, sobald ein Visualisierungsobjekt in ein Standardprojekt eingefügt wird. Sie referenzieren weitere Bibliotheken, die hier nicht aufgeführt sind. Im Standardfall müssen Sie die Visualisierungsbibltioheken nicht explizit einfügen oder in Ihren Applikationen verwenden:

VisuElems.library (Basis-Set an Visualisierungselementen)

VisuElemMeter (meter and bar display Elemente)

VisuElemWinControls (Tabelle, Textfeld, scrollbar elements)

VisuElemTrace (Trace Element)

VisuInputs.library (Handhabung der Eingaben auf eine Visualisierung)

8.13 Visualisierungsprofile

Jedes Visualisierungsprojekt, also ein Projekt, das mindestens ein Visualisierungsobjekt enthält, muss auf einem Visualisierungsprofil basieren. Dieses Profil definiert folgendes:

- die Namen und Versionen der Visualisierungsbibliotheken die automatisch im Projekt eingebunden werden sobald ein Visualisierungsobjekt angelegt wird
- eine Auswahl von aus den eingebundenen Bibliotheken stammenden Visualiserungselementen, die in der Werkzeug-Box des Visualisierungs-Editors bereitstehen sollen

Die **Profilkonfiguration** wird außerhalb eines Projekts im Visualisierungselemente-Repository Editor vorgenommen. Es können mehrere Profile definiert und auf dem lokalen System abgelegt werden.

Welches **Profil standardmäßig im Projekt verwendet wird**, ist in den Projekteinstellungen (Visualisierungsprofil) definiert. Dort kann auch jederzeit auf ein anderes Profil umgeschaltet werden. Beachten Sie, dass das gewählte Profil für alle Geräte und Applikationen gilt.

Wenn das Profil gewechselt wird, erscheint eine Meldung, dass dies möglicherweise ein Einloggen ohne Online Change oder Download verhindert.

Das Wechseln des Profils verursacht eine automatische Aktualisierung im Bibliotheksverwalter bezüglich der erforderlichen Bibltiotheken. Diese müssen nicht händisch angepaßt werden.

Ein Visualisierungsprofil muss mindestens eine **"Haupt"-Visualisierungsbibliothek** definieren, d.h. eine Bibliothek, die die Ersetzungen für die Platzhalter, die in anderen Visualisierungsbibliotheken verwendet sind, definiert. Im Falle einer Standardinstallation von 3S-Smart Software Solutions GmbH ist dies die Bibliothek **VisuElems.library**.

Wenn ein altes Projekt geöffnet wird, das noch nicht mit einem Visualisierungsprofil angelegt wurde, werden Sie gefragt, ob auf den neuen Profil-Mechanismus umgeschaltet werden soll oder nicht.

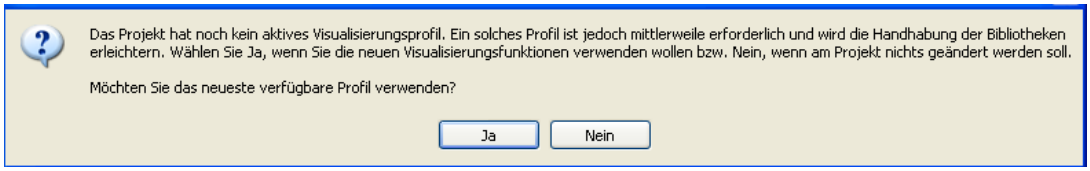


Abb. 47 Meldungsbox, falls Profil fehlt

Wenn Ja geantwortet wird, wird das "Neueste Profil" verwendet. Bei Nein wird das älteste verfügbare Profil in den Projekteinstellungen eingetragen (genannt "**Kompatibilitätsprofil**"), aber im Projekt werden keine weiteren Änderungen vorgenommen.

8.14 Target-Visualisierung

Der Code, der für die Visuailsierungsobjekte des Projekts erzeugt wird, kann auf die Steuerung, d.h. auf das Zielsystem (Target) geladen werden.

Wenn es vom Zielsystem unterstützt wird und wenn ein entsprechender Monitor verfügbar ist, kann die Visualisierung direkt auf dem Steuerungsrechner gestartet werden, ohne dass das Programmiersystem läuft. Dies ermöglicht eine erhebliche Einsparung an Speicherverbrauch.

Hinweis: Wenn das Programmiersystem läuft, kann die Visualisierung auch dort angezeigt werden. Dies ist allerdings nicht äquivalent mit der reinen "Diagnose-Visualisierung".

Die Target-Visualisierung wird automatisch auf der Steuerung gestartet, sobald dort die entsprechende Applikation gestartet wird, wenn... :

- eine Startvisualisierung definiert ist (siehe Visualization Manager)
- der Name der Applikation auf der Steuerung "<application>.app" lautet (diese Datei wird automatisch beim Download eines Visualisierungsprojekts angelegt)
- ein entsprechendes Laufzeitsystem gewählt wurde ("mit Target-Visu")

Hinweis: Die VISU_TASK, die automatisch in der Taskkonfiguration eingefügt wird, wenn ein TargetVisualization- oder WebVisualization-Objekt unter dem Visualization Manager eingehängt wird, wird wieder entfernt, wenn das letzte dieser Client-Objekte gelöscht oder in seinen Eigenschaften auf "Vom Übersetzen ausschließen" gesetzt wurde.

Einstellungen zur Darstellung einer Target-Visualisierung werden vom **Visualization Manager** im Unterobjekt Target-Visualisierung gehandhabt.

Beachten Sie im Falle von kleinen Zielsystemen die Möglichkeit, die Visualisierung integriert im Programmiersystem zu betreiben.

8.15 Web-Visualisierung

Die Web-Visualisierung kann verwendet werden wenn unter dem Objekt Visualization Manager ein Objekt Web-Visualisierung eingehängt ist. Dies ist nur möglich wenn das **Zielsystem** die Web-Visualisierung unterstützt. Dazu muss im Zielsystem ein spezieller **Web Server** vorhanden sein, der die Kommunikation zwischen Zielsystem und Web Client ermöglicht. Ob das Objekt Web-Visualisierung automatisch eingehängt wird oder manuell hinzugefügt werden muss, hängt von der Gerätebeschreibung des Zielsystems ab.

Die Web-Visualisierung ist als Java-Applet realisiert, welches beim Starten die Bildinformation (damit ist die Zeicheninformation der Visualisierung gemeint) vom WebServer anfragt. Danach werden nur noch Bildänderungen zyklisch übertragen. Bei einem **Download** eines Visualisierungsprojekts werden alle für die Web-Visualisierung benötigten Dateien in das Unterverzeichnis **/visu** des Laufzeitsystems übertragen. Dazu gehören das **Java-Applet**, die **Basis-HTML-Seite** (*.htm-Datei) der Visualisierung, sowie alle in der Visualisierung benötigten **Abbildungen**.

Voraussetzung für eine Web-Visualisierung:

- Zielsystem mit entsprechender Konfiguration und Web-Server
- Angabe der Startvisualisierung und des Namens der zu übertragenden htm-Datei im Visualization Manager
- Web-Browser: IExplorer oder Mozilla Firefox; plus Java-VRM ab V1.4

Hinweis: Die VISU_TASK, die automatisch in der Taskkonfiguration eingefügt wird, wenn ein TargetVisualization- oder WebVisualization-Objekt unter dem Visualization Manager eingehängt wird, wird wieder entfernt, wenn das letzte dieser Client-Objekte gelöscht oder in seinen Eigenschaften auf "Vom Übersetzen ausschließen" gesetzt wurde.

Konfiguration im Programmiersystem

Die Einstellungen für eine Web-Visualisierung bzgl. Datenübertragung, Darstellungsgröße, zu verwendende Startvisualisierung und htm-Datei sind im Visualization Manager im Unterobjekt Web-Visualisierung vorzunehmen.

Konfiguration des Zielsystems

Gegebenenfalls muss die Web-Visualisierungsfunktionalität im Zielsystem erst noch konfiguriert werden. Folgende Einträge müssen in der Konfigurationsdatei <Zielsystem>.cfg vorhanden sein:

```
[ComponentManager]
Component.<n>=CmpWebServerHandlerV3
Component.<n>=CmpWebServer
[CmpWebServer]
FileUploadDirectory=C:\Programme\ PLC Designer \GatewayPLC\visu
```

FileUploadDirectory muss den Pfad des Laufzeitsystems beschreiben, in dem der WebServer verfügbar ist. An den Pfad muss Unterverzeichnis "\visu" angehängt werden. Dorthin werden alle für die Web-Visualisierung benötigten Dateien übertragen.

Aufruf einer Web-Visualisierung über Internet

Geben Sie im Web-Browser folgende Adresse ein:

```
http://<IP_Adresse des WebServers>:<Port des WebServers>/<webvisu>.htm
```

Beispiel:

```
http://localhost:8080/webvisu.htm
```

<webvisu>.htm ist die im Visualization Manager definierte HTML-Startseite der Visualisierung. Über sie wird nach dem Aufruf im Browser zunächst die Startvisualisierung dargestellt, die ebenfalls im Manager definiert ist . Danach kann die Visualisierung im Browser bedient werden.

8.16 Visualisierung läuft im Programmiersystem

Zu Diagnosezwecken kann es gewünscht sein, eine die einer Applikation zugehörige Visualisierung(en) nur innerhalb des Programmiersystems laufen zu lassen, ohne Visualisierungscode auf die Steuerung laden zu müssen.

Diese "**Diagnose-Visualisierung**", die **ab V3.3.2.0** verfügbar ist, wird **automatisch** verwendet, wenn kein Client-Objekt "TargetVisualization" oder "WebVisualiaztion" im Gerätebaum unterhalb des Visualization Manager eingehängt ist. Dann wird **kein Visualisierungscode** erzeugt und auf die Steuerung geladen. Dies bedeutet allerdings einige Einschränkungen, die im folgenden aufgelistet sind.

Wenn Sie nicht die TargetVisualization und/oder WebVisualization Objekte unterhalb des Visualization Manager entfernen wollen, um den Diagnose-Modus der Visualisierung zu erhalten, können Sie alternativ auch die Client-Objekte vom Übersetzen aussschließen, indem Sie die Einstellung "Vom Übersetzen ausschließen" in den jeweiligen Objekteigenschaften aktivieren.

Hinweis: Wenn in einem Projekt, das ursprünglich mit einer Version < V3.3.2.0 erstellt wurde, eine feste Compiler-Version eingestellt wurde, dann verhält sich das Projekt entsprechend den Möglichkeiten dieser Version! Vor V3.3.2.0 mußte die Einstellung "Die SPS nicht für Visu verwenden" aktiviert werden, um eine Visualisierung nur im Programmiersystem laufen zu lassen.

Numerische Werte, die in einer Diagnose-Visualisierung in einem Element angezeigt werden ("%s" in der "Text"-Eigenschaft), werden entsprechend des aktuell gesetzten Darstellungsformats (binär, dezimal, hexadezimal) angezeigt.

Hinweis: Im Unterschied zur Target-Visualisierung verhalten sich in der Diagnose-Visuailsierung VAR_INPUT Variablen wie VAR_IN_OUTs.

Einschränkungen

Ausdrücke, Monitoring

Die Diagnose-Visualisierung unterstützt nur Ausdrücke, die vom Monitoring-Mechanismus des Programmiersystems gehandhabt werden können. Diese sind:

- normale Variablenzugriffe wie PLC_PRG.myPou.nCounter
- komplexe Zugriffe wie im folgenden aufgelistet; man beachte, dass für solche ein Laufzeitsystem V3.3 SP2 oder neuer erforderlich ist!:
  - Zugriff auf ein Array von skalaren Datentypen, wobei eine Variable als Index verwendet wird (a[i])
  - Zugriff auf ein Array von komplexen Datentypen (Strukturen, Funktionsbausteinen, Arrays) wobei eine Variable als Index verwendet wird (a[i].x)
  - Zugriff auf ein mehrdimensionales Array von allen Arten von Datentypen, mit einer oder mehreren Variablen-Indizes (a[i, 1, j].x)
  - Zugriff auf ein Array mit konstantem Index [a[3])
  - Zugriffe wie die oben beschriebenen, in denen einfache Operatoren für die Berechnungen innerhalb der Indexklammer verwendet werden (a[i+3])
  - verschachtelte Kombinationen der oben gelisteten komplexen Ausdrücke (a[i + 4 * j].aInner[j * 3].x)
- in Index-Berechnungen unterstützte Operatoren: +, -, *, /, MOD
- Pointer-Monitoring wie p^.x
- nicht unterstützt werden Methoden oder Funktionsaufrufe außer den folgenden: alle Standard-Stringfunktionen, alle Typkonvertierungsfunktionen wie INT_TO_DWORD, alle Operatoren wie SEL, MIN, ...

Eingaben:

Innerhalb der Eingabeaktion "ST-Code ausführen" wird nur eine Liste von Zuweisungen unterstützt.

Beispiel:

```
PLC_PRG.n := 20 * PLC_PRG.m;
// nicht erlaubt
IF PLC_PRG.n < MAX_COUNT THEN
   PLC_PRG.n := PLC_PRG.n + 1;
END_IF

//statt dessen folgendes verwenden:

PLC_PRG.n := MIN(MAX_COUNT, PLC_PRG.n + 1);
```

Hinweis: Wenn eine Liste von Zuweisungen verwendet wird, wird der Wert auf der linken Seite erst im nächsten Zyklus zugewiesen. Eine unmittelbar folgende Verarbeitung in der nächsten Zeile ist nicht möglich.

Visualisierungsschnittstelle:

Innerhalb der Schnittstellendefinition einer Visualisierung darf der Typ "Schnittstelle" ("INTERFACE") nicht verwendet werden.

DataServer:

Die Diagnose-Visualisierung - da diese nur dafür gedacht ist, die aktuelle Applikation passiv zu betrachten - zeigt im Unterschied zur Target-Visualisierung die laufenden Werte von Variablen, die über den Data Server aus einer anderen Datenquelle geliefert werden, nicht an! Für diese werden nur die Initialwerte oder die zuletzt übertragenen dargestellt.

8.17 Visualisierung auf externen Geräten - HMI

Durch Benutzen des PLC Designer-HMI (HMI=Human Machine Interface) kann eine Visualisierung, die mit dem PLC Designer-Programmiersystem erzeugt wurde, auf einem anderen Gerät laufen als das, auf welchem die visualisierte Applikation abgearbeitet wird.

In Unterschied zu dem in der Version V2.x integrierten Version des PLC Designer-HMI ist es nicht länger erforderlich das komplette Projekt auf das HMI herunterzuladen. Aktuell ist das PLC Designer-HMI ein **Laufzeitsystem**, welches den Client, eine Visualisierungsapplikation und einen Data Server umfasst. Folglich ist die Benutzung des PLC Designer-HMI plattformunabhängig (Windows, CE, Linux, QNX, ...).

Aufgrund der internen Behandlung des HMI als Target-Visualisierung besitzt es eine hohe Leistungsfähigkeit.

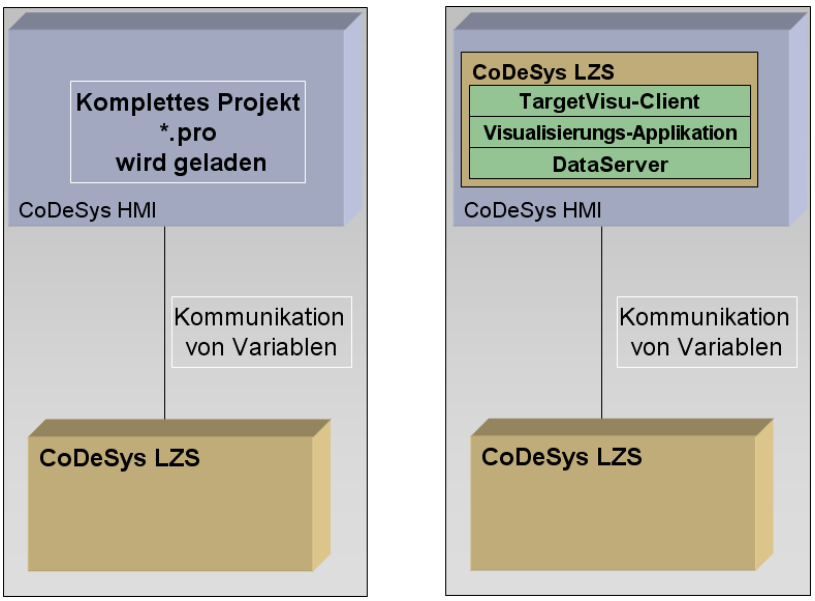


Abb. 48 Konzept des PLC Designer-HMI - Version V2.x (links) gegenüber Version V3.x (rechts)

Nachdem das PLC Designer-HMI auf dem entsprechenden Gerät gestartet wurde, wird die zugehörige Visualisierung (definiert im Visualization Manager) dort im Vollbildmodus gestartet. Mittels der Tastatur / der Maus kann der Benutzer die in der Visualisierung enthaltenen Kontroll- und Überwachungsfunktionen steuern. Diese Funktionen werden realisiert mit Hilfe von Visualisierungselementen, die im Online-Modus bedient werden können. Daher bilden sie ein Interface zur visualisierten Applikation, über welches das Verhalten des zugehörigen Projekts beeinflusst werden kann.

Installation:

PLC Designer-HMI wird zusammen mit dem Standard PLC Designer-Setup installiert. Für Testzwecke steht eine zeitlimitierte zur Verfügung.

Start:

Im Prinzip ist das PLC Designer-HMI eine Target-Visualisierung und kann daher auf einer beliebigen Plattform gestartet werden. Speziell auf einer Windows Plattform wird das PLC Designer-HMI (PLCDesignerSPHMI.exe) vom Startmenü aus aufgerufen.

Abb. 49 Starten von PLC Designer-HMI aus dem Startmenü

Visualisierung:

Um eine Visualisierung auf einem externen Gerät zu betreiben, muss das **Gerät "PLC Designer HMI for Win32"** in den Gerätebaum des zugehörigen Projekts eingefügt werden. Dies Gerät ist eine eingeschränkte Version der Steuerung "PLC Designer Win NT", die nur für Visualisierungszwecke gedacht ist, aber beispielsweise kein Input/Output handhaben kann.

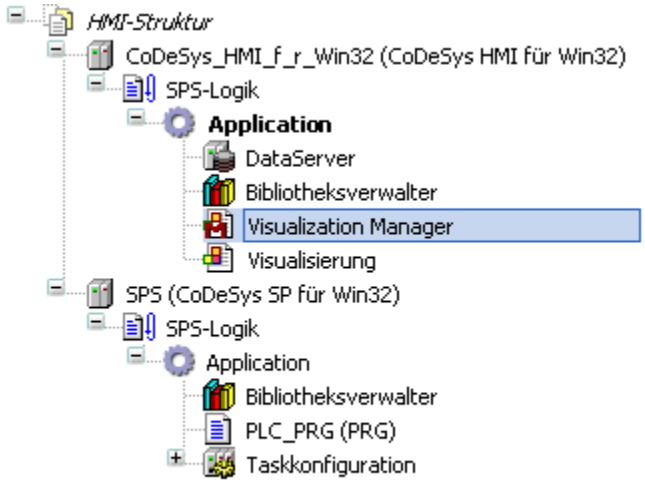
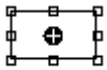
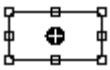


Abb. 50 Beispiel eines Projekts, das PLC Designer-HMI zur Visualisierung nutzt

Unterhalb der zugehörigen Applikation muss ein Data Server geeignet konfiguriert werden um diejenigen Parameter zur Verfügung zu stellen, welche in der Applikation enthalten sind und in die Visualisierung einfließen sollen. Die Visualisierung selbst ist auf übliche Weise implementiert. Stellen Sie sicher, dass Sie beim Konfigurieren des zugehörigen Visualisierungs-Managers das Kontrollkästchen "Target-Visualisierung öffnen" aktivieren.

8.18 Visualisierungselemente vom Typ Visualisierung

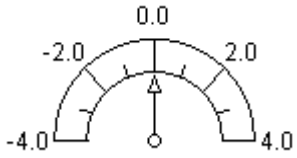
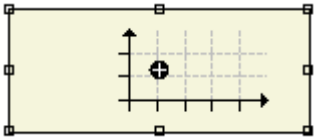
Die Kategorie "Visualisierung" des Werkzeuge-Fensters im Visualierungs-Editor enthält die folgenden Elemente:

Rectangle		Rechteck, Abgerundetes Rechteck und Ellipse gehören zum gleichen Elementtyp. Sie können durch Ändern der Type-Eigenschaft ineinander konvertiert werden.
Rounded Rectangle		
Ellipse		
Line		Einfache Linie: Die Neigung eines "Line"-Elements kann zwischen "oben links - unten rechts" und "unten links - oben rechts" umgeschaltet werden.

Polygon		Polygon, Polylinie und Kurve gehören zum gleichen Elementtyp. Sie können durch Ändern der Type-Eigenschaft ineinander konvertiert werden.
Polyline		Ein zusätzlicher Positionspunkt, der die Abschnitte und die Form eines Elements bestimmt, kann eingefügt werden, indem bei gedrückter <Strg>-Taste auf einen vorhandenen Punkt (kleines Rechteck auf Rahmenlinie) geklickt wird. Ein Punkt kann gelöscht werden, indem bei gedrückten <Umsch>+<Strg>-Tasten auf den Punkt geklickt wird. In einem Kurvenelement dienen die neben der Linie liegenden Punkte als "Griffe", um die Kurvenform zu verändern.
Curve		
Frame		Rahmen: Ein Rahmen (Frame), wie "Frames", die z.B. aus Internetseiten bekannt sind, definiert einen Teilbereich in der aktuellen Visualisierung, dem eine oder mehrere andere Visualisierungen zugewiesen sind, von denen eine im Online-Modus angezeigt wird. Grundsätzlich kann ein Frame wie ein Rechtecke-Element konfiguriert werden. Ein Wechsel der angezeigten Visualisierungen kann bei entsprechender Konfiguration durch eine Benutzereingabe erfolgen. Die Zuweisung der Visualisierungen zu einem Frame erfolgt über den 'Konfiguration der Framevisualisierungen', der über Befehl 'Frame Auswahl' geöffnet wird. Hier werden alle verfügbaren Visualisierungen angeboten und die gewünschten können ausgewählt werden. Bezüglich der Konfiguration der Visualisierungwechsel durch Benutzereingaben siehe die möglichen Mausaktionen.
Button		Einem Schaltflächen-Element kann ein Bild und eine "Höhe" (die die Relief-Ansicht bewirkt) zugewiesen werden.
Image		Ein Bild-Element wird durch eine Bilddatei gefüllt, die über ihre ID und ggfs. den Namen der Bildersammlung, in der sie verwaltet wird, definiert wird. Die Angabe der Datei kann auch dynamisch konfiguriert werden, d.h. über eine Projektvariable. Siehe hierzu: Bildersammlungen.

8.19 Visualisierungselemente vom Typ Complex Controls

Die Kategorie "Complex Controls" des Werkzeuge-Fensters im Visualisierungs-Editor enthält die folgenden Elemente:

Meter		Dieses Element fügt einen Tachometer in die Visualisierung ein, dessen minimaler und maximaler Anzeigewert angegeben werden können. Die Position der Nadel gibt den aktuellen Wert der verknüpften Eingabevariable wieder. Für bestimmte Wertebereiche können spezielle Hintergrundfarben definiert werden.
Bar Display		Dieses Element fügt eine Balkenanzeige in die Visualisierung ein, deren minimaler und maximaler Anzeigewert angegeben werden können. Die Länge des Balkens gibt den aktuellen Wert der verknüpften Eingabevariable wieder. Für bestimmte Wertebereiche können spezielle Farben definiert werden. Standardmäßig wird ein horizontaler Balken angezeigt; nach Einfügen des Elements kann die Orientierung jedoch innerhalb der Elementeigenschaften in einen vertikalen Balken verändert werden.
Trace		Dieses Element erlaubt das Einbinden eines Trace in die Visualisierung. In seinen Elementeigenschaften wird der Name des darzustellenden Trace angegeben. Die Konfiguration der aufgezeichneten Variablen wird jedoch innerhalb der Konfiguration des Trace selbst vorgenommen.

8.19.1 Konfiguration des Elements Meter

Das Visualisierungselement Meter wird einer Visualisierung mittels drag&drop aus dem Werkzeuge-Fenster hinzugefügt. Sein wie ein Tachometer gestaltetes Aussehen erlaubt die Visualisierung des aktuellen Werts der verknüpften Eingangsvariable.

Sehen Sie im Folgenden ein Beispiel für das Einfügen und die Konfiguration eines Meter Elements:

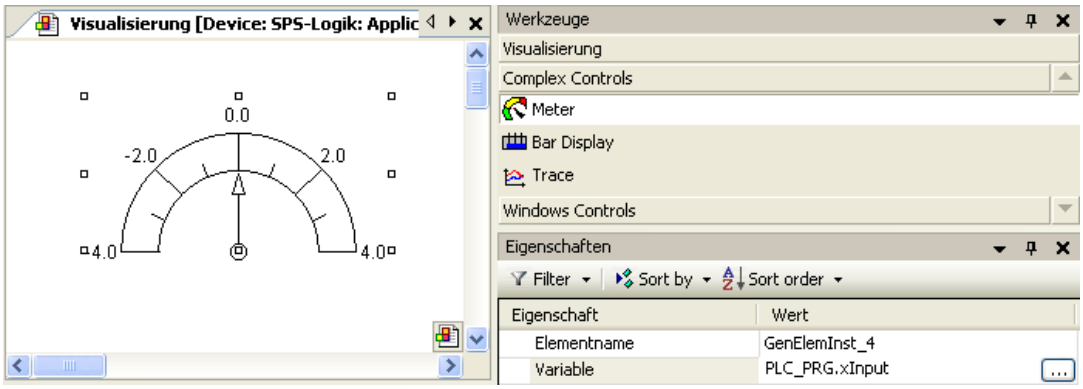


Abb. 51 Beispiel - Einfügen des Meter Elements und Verknüpfung mit Eingangsvariablen

Die zu verknüpfende Eingangsvariable - im Beispiel eine Variable namens "xInput"- muss innerhalb der Elementeigenschaften des Meter Elements angegeben werden; nach einem Klick in das Eingabefeld steht der Schalter [...] zur Verfügung, welcher betätigt werden kann um das Projekt nach der Eingangsvariable zu durchsuchen. Achten Sie darauf die Eingangsvariable durch Angabe ihres vollständigen Pfads im Gerätebaum zu spezifizieren.

Die Orientierung und Größe der Skalenanzeige wie auch Farbe und Aussehen des Pfeil kann im Abschnitt "Arrow" der Elementeigenschaften festgelegt werden.

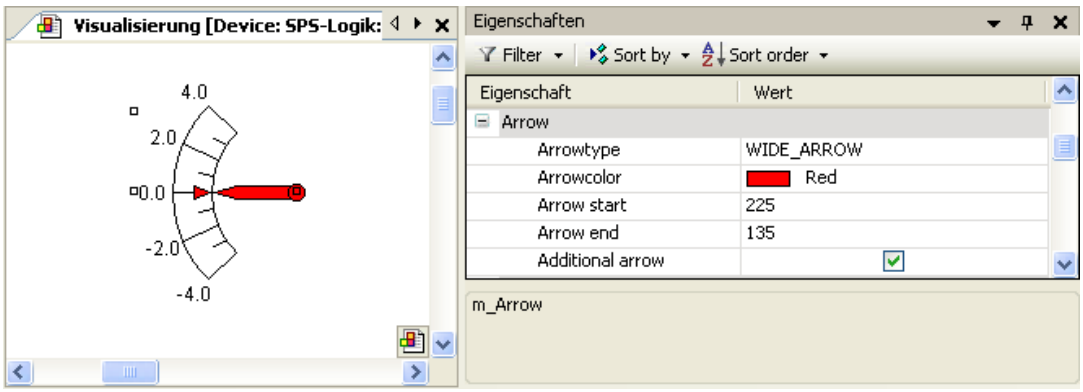


Abb. 52 Beispiel - Konfiguration innerhalb des Abschnitts "Arrow" in den Elementeigenschaften

Die Zahl im Eingabefeld "Arrow start"/"Arrow end" gibt den Winkel (in Grad) an, dem der linke/rechte Rand der Skala mit der horizontalen Gerade einschließt. Dieser Winkel ist mathematisch, d.h. entgegen dem Uhrzeigersinn orientiert, so dass der Wert im Feld "Arrow start" immer größer sein muss als der Wert von "Arrow end"! Das Paar aus Anfangs- und Endwert ist periodisch mit 360 Grad.

Im Abschnitt "Scale" kann der Zahlenbereich für die Skala und eine Grob-/ und Feinskalierung bestimmt werden.

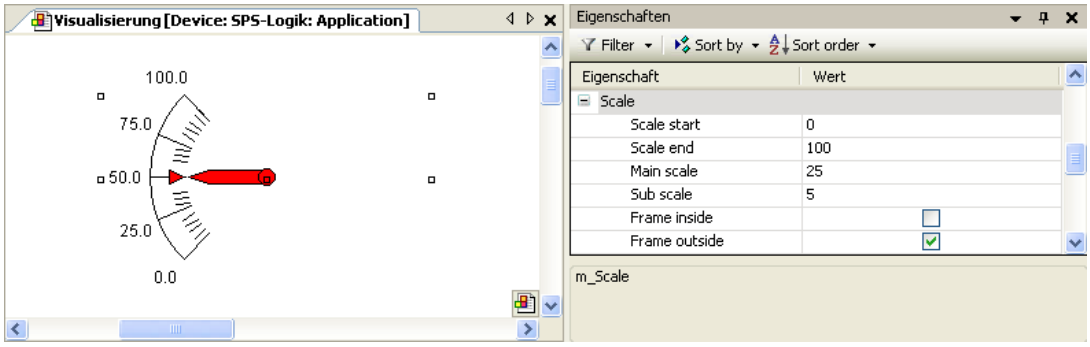


Abb. 53 Beispiel - Konfiguration innerhalb des Abschnitts "Scale" in den Elementeigenschaften

Der Wert von "Scale start" wird am linken Rand der Skala eingetragen. Daher muss er kleiner sein als der Wert von "Scale end", der am rechten Rand der Skala eingefügt wird. Im Gegensatz zu der groben Skalierung kann die feinere Unterteilung ("Sub scale") weggelassen werden, indem man den Wert für ihre Abstände auf 0 setzt. In diesem Fall würden also keine feinen Skalenstriche angezeigt werden. Da das Kontrollkästchen "Frame inside" im Beispiel deaktiviert wurde, ist der innere Kreisbogen der Skala ausgeblendet.

Nachdem die Skalierung festgesetzt wurde, soll nun im Abschnitt "Label" die Beschriftung der Skala formatiert werden.

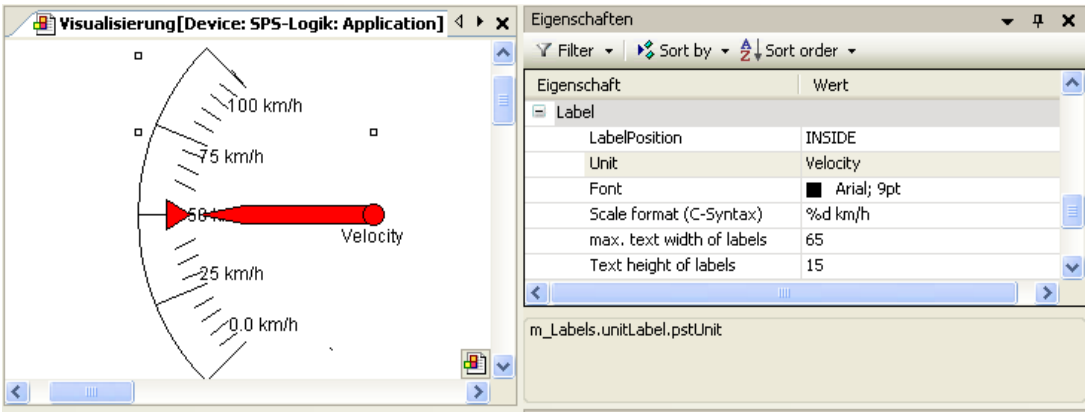


Abb. 54 Beispiel - Konfiguration innerhalb des Abschnitts "Label" in den Elementeigenschaften

Durch Abänderung der "LabelPosition" von OUTSIDE in INSIDE verlagern wir die Anzeige der Skalierungsmarken in das Kreisinnere. Der Eintrag im Feld "Unit" erscheint unterhalb des Fußpunkt des Pfeils. Nach Auswahl einer geeigneten Schrift(farbe), passen wir die Formatierung der Skalierungsmarken an: der Zahlenwert der Skalierung ist gemäß der Syntax der Programmiersprache C zu formatieren (benutzen Sie %d für ganze Zahlen und %.Xf für Gleitkommazahlen, wobei X zu ersetzen ist durch die gewünschte Anzahl an Nachkommastellen). Die Werte in den folgenden beiden Eingabefeldern werden entsprechend Ihrer vorhergehenden Eintellungen in diesem Abschnitt eingetragen. Sie müssen diese Werte nur dann verändern, falls die automatische Anpassung nicht zum gewünschten Ergebnis führt.

Schließlich können im Abschnitt "Colors" bestimmte Bereich der Skala noch eingefärbt werden indem man sog. "color areas" anlegt mit Hilfe der Taste Neu erstellen. Die Farbbereiche werden in aufsteigender Ordnung durchnummeriert. Jeder Farbbereich wird innerhalb der Elementeigenschaften mit eigenen Eingabefeldern ausgestattet.

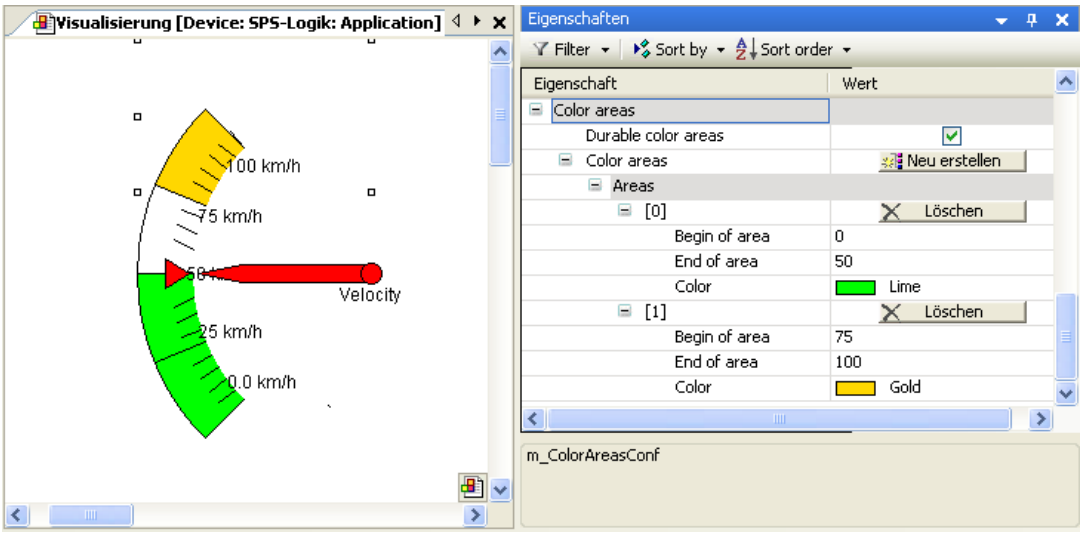


Abb. 55 Beispiel - Konfiguration innerhalb des Abschnitts "Colors -Color areas" in den Elementeigenschaften

Die Felder "Begin of area" und "End of area" stehen zur Spezifizierung des Unterbereichs der Skalierung zur Verfügung. Die Einfärbung kann aus einem Aufklappmenü ausgewählt werden und mit Hilfe des Schalters Löschen können einmal angelegte Farbbereiche wieder gelöscht werden.

Der Effekt des Kontrollkästchens "Durable color areas" ist nur im Online Modus ersichtlich:

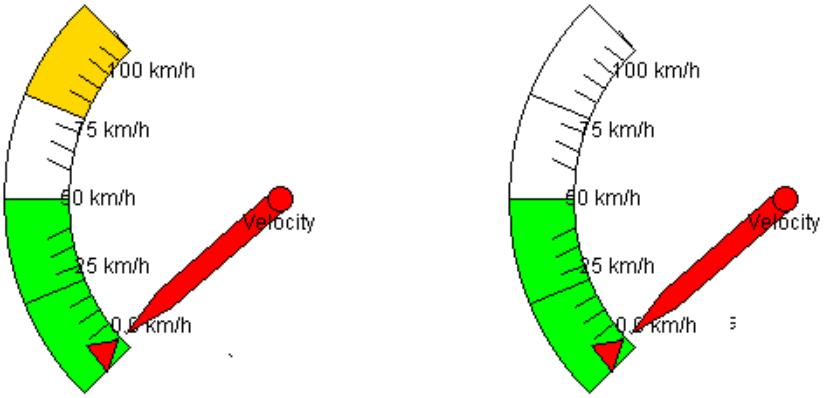


Abb. 56 Beispiel - Meter Element im Online Modus

8.19.2 Konfiguration des Elements Bar Display

Das Visualisierungselement Bar Display kann zur Visualisierung durch Ziehen aus dem Werkzeug-Fenster eingefügt werden. Seine Gestalt gleicht einem Thermometer und erlaubt somit den aktuellen Wert der verknüpften Eingangsvariable anzuzeigen.

Sehen Sie im Folgenden ein Beispiel für das Einfügen und die Konfiguration eines Bar Display Elements:

Abb. 57 Beispiel - Einfügen des Bar Display Elements und Verknüpfung mit Eingangsvariablen

Die zu verknüpfende Eingangsvariable - im Beispiel eine Variable namens "xInput"- muss innerhalb der Elementeigenschaften des Bar Display Elementes angegeben werden; nach einem Klick in das Eingabefeld der Eigenschaft "Variable" steht Ihnen der Schalter ... zur Verfügung, welcher benutzt werden kann um im Projekt nach der Variable zu browsen. Achten Sie darauf die Eingangsvariable durch Angabe ihres vollständigen Pfads im Gerätebaum zu spezifizieren.

Die Orientierung und Platzierung des Balken in Bezug auf die Skalierung kann innerhalb des Abschnitts "Bar" der Elementeigenschaften eingestellt werden.

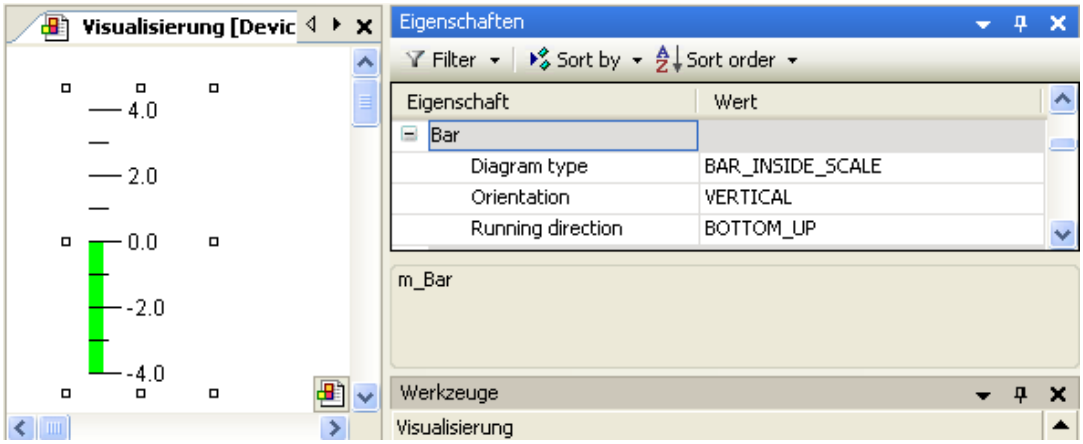


Abb. 58 Beispiel - Konfiguration innerhalb des Abschnitts "Bar" in den Elementeigenschaften

Im Beispiel wurde die Orientierung des Balken von HORIZONTAL auf VERTICAL geändert. Diese Einstellung ändert auch die möglichen Angaben für die "Running direction": anstelle von LEFT_RIGHT und RIGHT_LEFT, also der Verlauf des Balkens von links nacht rechts oder vice versa, müssen wir nun zwischen BOTTOM_UP (für einen aufsteigenden Balken) oder TOP_DOWN (für einen absteigenden Balkenverlauf) wählen. Die Position des Balken in Bezug zur Skalierung kann durch Einstellung des "Diagram type" bestimmt werden.

Der Abschnitt "Scale" ist vorgesehen zur Festlegung des Wertebereichs der Skala und groben bzw. feineren Unterteilung der Skalierung.

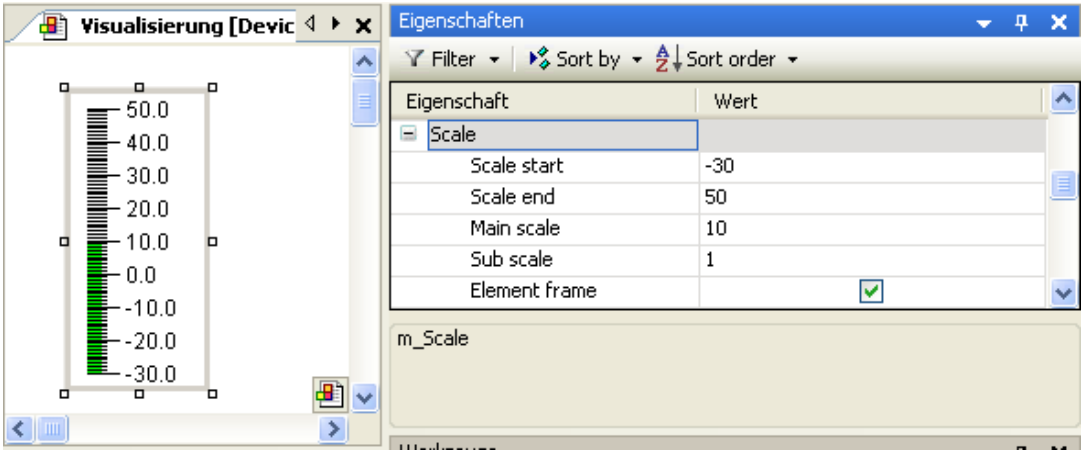


Abb. 59 Beispiel - Konfiguration innerhalb des Abschnitts "Scale" in den Elementeigenschaften

Der Wertebereich der Skala wird von unten durch den Wert in "Scale start" und von oben durch den Wert in "Scale end" begrenzt. Der Wert von "Scale start" muss also geringer sein als der von "Scale end". Beide Distanzangaben in "Main scale" wie "Sub scale" können auf 0 gesetzt werden um eine Anzeige der Skalierungsstriche auszuschalten: wird der Wert der Grobskalierung (Main scale) auf 0 gesetzt, so werden unabhängig vom Wert der Feinskalierung keinerlei Skalierungsstriche gemalt. Wird die Feinskalierung (Sub scale) auf 0 gesetzt, so werden nur die Skalierungstriche der groben Unterteilung gezeichnet. Da wir im Beispiel einen Rahmen um das Element gezeichnet haben möchten, aktivieren wir das Kontrollkästchen "Element frame".

Der Abschnitt "Label" dient zur Beschriftung des Balken:

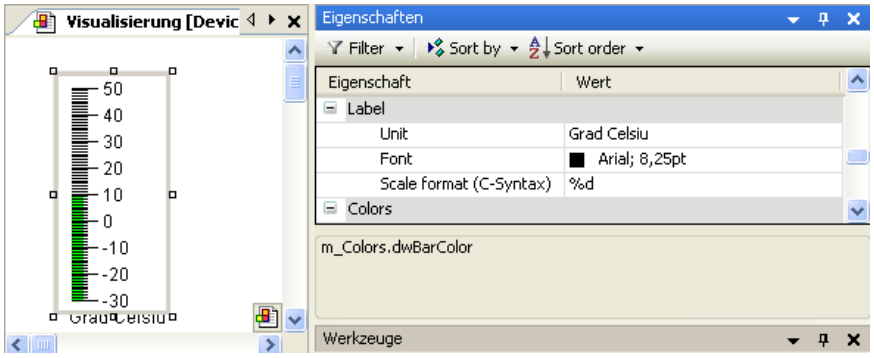


Abb. 60 Beispiel - Konfiguration innerhalb des Abschnitts "Label" in den Elementeigenschaften

Der Eintrag im Eingabefeld "Unit" ist zur Angabe der Einheit gedacht und wird zentriert unterhalb des Balken angezeigt. Nach Wahl einer geeigneten Schrift(farbe) passt man das Format der Beschriftungen an: die Angabe der Formatierung für Zahlenwerte muss entsprechend der Syntax der Programmiersprache C vorgenommen werden (benützen Sie %d für ganzzahlige Werte und %.Xf für Gleitpunktzahlen, wobei X durch die gewünschte Anzahl an Nachkommastellen ersetzt werden muss).

Schließlich kann die Einfärbung des Elements innerhalb des Abschnitts "Colors" gesetzt werden:

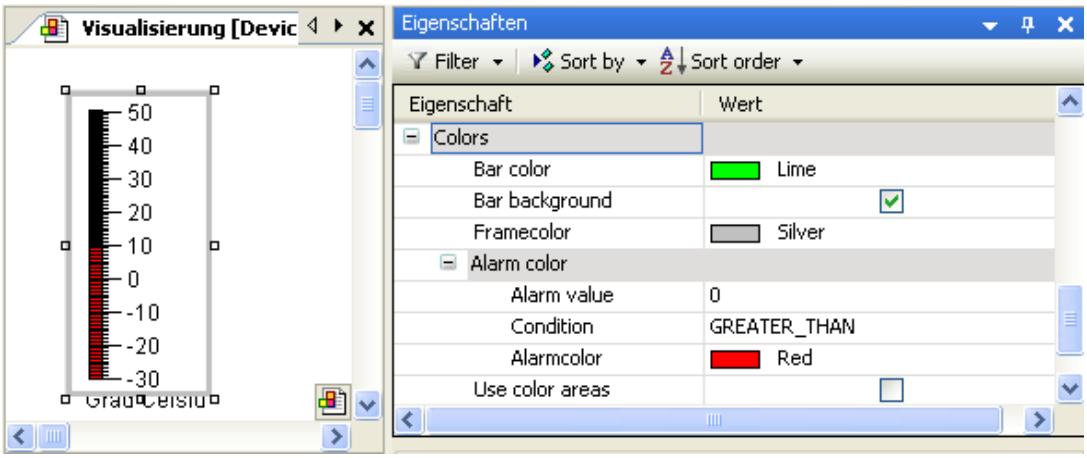


Abb. 61 Beispiel - Konfiguration innerhalb des Abschnitts "Colors" in den Elementeigenschaften

Zuerst kann die Farbe des Balken selbst bestimmt werden (Bar color). Standardmäßig wird der Hintergrund des Balkens, das ist der Teil der Balkenzeile, die aktuell nicht vom Balken ausgefüllt ist, weiß angezeigt. Bei Aktivierung des Kontrollkästchens "Bar background" kann die Hintergrundfarbe in schwarz geändert werden. Desweiteren kann die Farbe des Rahmens spezifiziert werden (Framecolor).

Innerhalb des Unterbereichs "Alarm color" kann eine Alarmfarbe angezeigt werden, in welcher der Balken angezeigt wird, sobald der aktuelle Wert der Eingangsvariable die Alarmbedingung erfüllt. Diese Bedingung ergibt sich aus der Definition des Alarmwerts (Alarm value), welcher entweder nicht überschritten (beim Setzen von "Condition" auf GREATER_THAN) oder nicht unterschritten werden darf (beim Setzen von"Condition" auf LESS_THAN).

Diese Einstellungen sind nicht länger von Belang, falls sogenannte Farbbereiche (color areas) angelegt und durch Aktivieren des Kontrollkästchens "Use color areas" eingeschaltet werden:

Teilbereichen der Skala kann dann eine bestimmte Farbe zugewiesen werden, indem für sie mit Hilfe des Schalters Neu erstellen ein Farbbereich erzeugt wird. Die Farbbereiche werden aufsteigend nummeriert. Jeder Farbereich wird innerhalb der Elementeigenschaften mit eigenen Eingabefeldern ausgestattet:

In "Begin of the area" und "End of the area" können die Grenzen des Unterbereichs festgelegt werden. Seine Einfärbung kann aus dem Aufklappmenü ausgewählt werden. Ein einmal angelegter Farbbereich kann durch einen Klick auf den zugehörigen Schalter Löschen gelöscht werden.

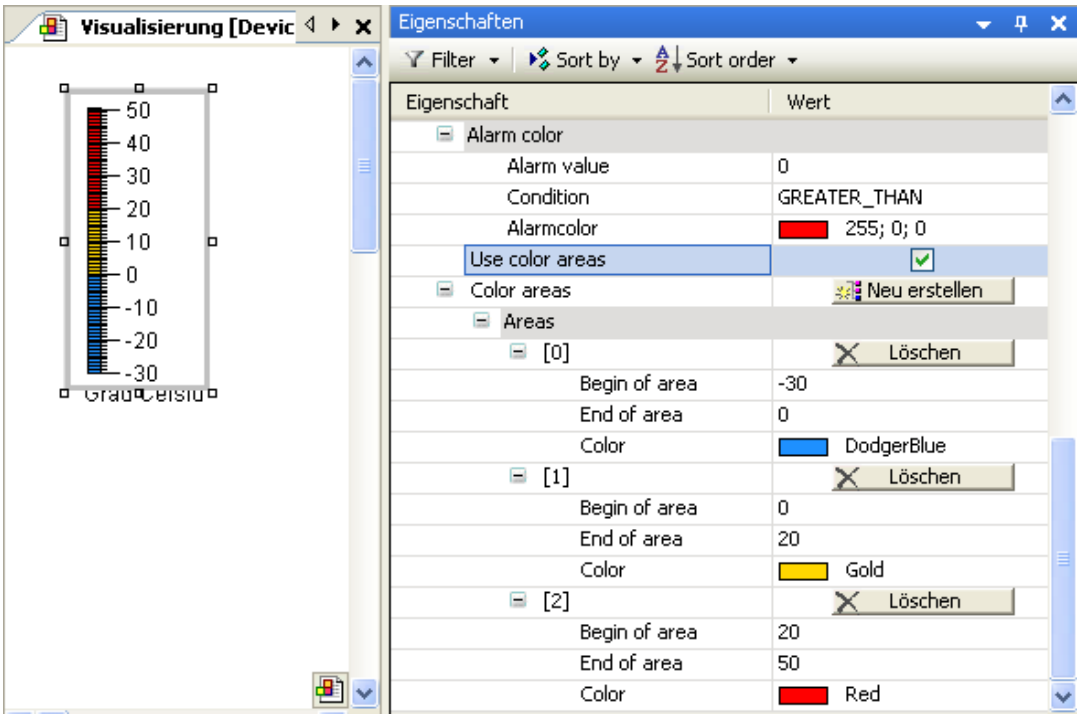


Abb. 62 Beispiel - Konfiguration innerhalb des Abschnitts "Colors - Color Areas" in den Elementeigenschaften

8.19.3 Konfiguration des Elements Trace

Um getracete Variablen innerhalb einer Visualisierung darstellen zu können, muss zunächst ein passendes Trace Objekt in das Projekt eingefügt werden. Darin wird die Konfiguration der aufzuzeichnenden Variablen, einer evetuellen Trigger-Variablen und des Trace selbst (Achsen, Aktualisierungsrate, etc.) vorgenommen. Die Aktivierung der Option "Trace-Baustein für Visualisierung anlegen" innerhalb der Trace-Konfiguration hat die implizite Erzeugung des Bausteins "<Tracename>_<Recordname>_VISU" zur Folge. Das ist Voraussetzung dafür, dass das Trace-Objekt als Eingang für das Visualisierungselement Trace dienen kann.

Das **Visualisierungselement "Trace"** kann durch Drag&Drop aus der Werkzeuge-Box in die Visualisierung eingefügt werden.

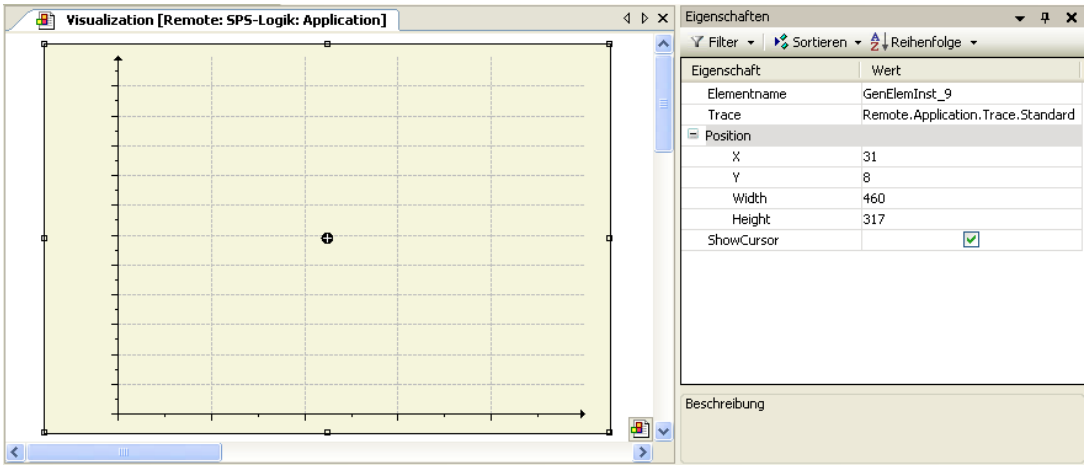


Abb. 63 Beispiel: Einfügen und Konfiguration des Visualisierungselements Trace

Das zugehörige Trace-Objekt wird innerhalb der Elementeigenschaften definiert: geben Sie hierzu den vollständigen Namen des Trace an, d.h. inklusive seines Pfades innerhalb des Gerätebaums! Eine Beschreibung der Option "ShowCursor" finden Sie weiter UNTEN.

Um die Visualisierung des Trace im Online-Modus beeinflussen zu können, mag das Hinzufügen weiterer Visualisierungselemente hilfreich sein. Die folgende Vorlage beinhaltet vier Schaltflächen zur Handhabung des Trace-Verhaltens in der Visualisierung:

Beispiel:

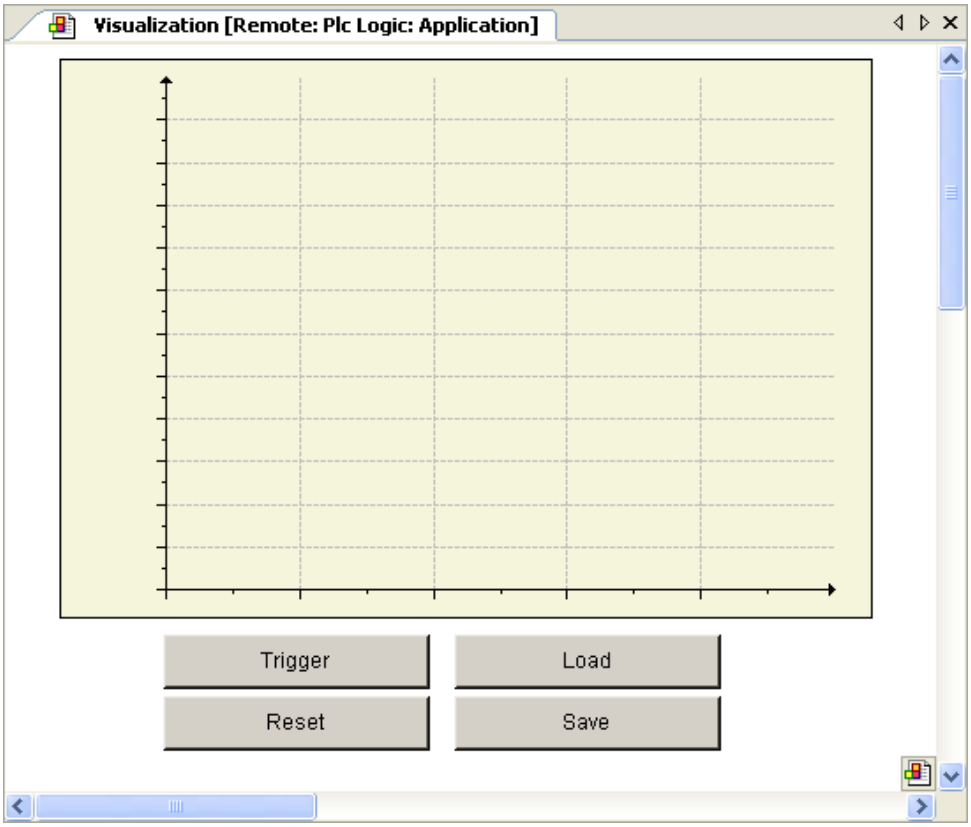


Abb. 64 Beispiel: Zusätzliche Elemente zur Handhabung des Trace

Die **Schaltfläche mit der Aufschrift "Trigger"** aktiviert das Triggern des Traces, d.h. der zugehörigen Trigger-Variable wird der Wert "TRUE" zugewiesen. Das wird innerhalb der Elementeigenschaften der Schaltfläche realisiert (Überbegriff "Eingaben"):

Eigenschaft	Wert
Eingaben	
OnDialogClosed	Konfigurieren...
OnMouseClick	Konfigurieren...
OnMouseDown	Konfigurieren...
ST-Code ausfü...	PLC_PRG.bTrigger := TRUE;
OnMouseEnter	Konfigurieren...
OnMouseLeave	Konfigurieren...
OnMouseMove	Konfigurieren...
OnMouseUp	Konfigurieren...

Die spezielle Aktion "OnMouseDown" ist verbunden mit der Ausführung eines ST-Codes, der die zugehörige Trigger-Variable auf TRUE setzt, hier "PLC_PRG:bTrigger:=TRUE;".

Entsprechend bewirkt die **Schaltfläche "Reset"** den Reset des Traces, indem die Trigger-Variable auf FALSE zurückgesetzt wird (hier "PLC_PRG.bTrigger:=FALSE;") und die zugehörige Reset-Variable auf TRUE gesetzt wird ("PLC_PRG.bReset:=TRUE;").

Die dritte **Schaltfläche mit der Aufschrift "Load"** erlaubt eine Modifikation des dargestellten Traces ohne bedingtes Ausloggen aus der zugehörigen Applikation. Dazu wurde ein auszuführender ST-Code angehängt, der die Trigger-Variable auf FALSE setzt. Des weiteren wird darin die Variable bLoad auf TRUE gesetzt, beispielsweise durch "Trace_MainTask_VISU.bLoad:=TRUE;". Diese boolesche Variable ist Teil des implizit erzeugten Bausteins "<Tracename>_<Recordname>_VISU" und kontrolliert das Laden einer Trace-Konfigurationsdatei mit der standardmäßigen Endung .tcg. Die Trace-Visualisierung (und natürlich auch der Trace selbst) werden auf die in dieser Datei gegebenen Konfiguration angepasst. Eine detaillierte Beschreibung der Konfigurationsdatei finden Sie WEITER UNTEN.

Die **Schaltfläche "Save"** erlaubt das Abspeichern der aktuell in der Visualisierung angezeigten Trace-Daten. Dazu wird im angehängten ST-Code nicht nur die Trigger-Variable auf FALSE gesetzt, sondern auch die Variable bSave auf TRUE (bspw. durch "Trace_MainTask_VISU.bSave:=TRUE;"). Diese boolsche Variable ist Teil des implizit generierten Bausteins "<Tracename>_<Recordname>_VISU" und veranlasst die Erzeugung einer Datendatei <Tracename>_<Recordname>.dta im Verzeichnis des Laufzeitsystems. Diese Datei enthält die Aktuell angezeigten Werte der aufgezeichneten variablen und kann bei Bedarf zu einem späteren Zeitpunkt wieder in den Trace eingelesen werden.

Die folgende Abbildung zeigt die Visualisierung eines Trace im Online-Modus nach Betätigen der Schaltfläche "Trigger":

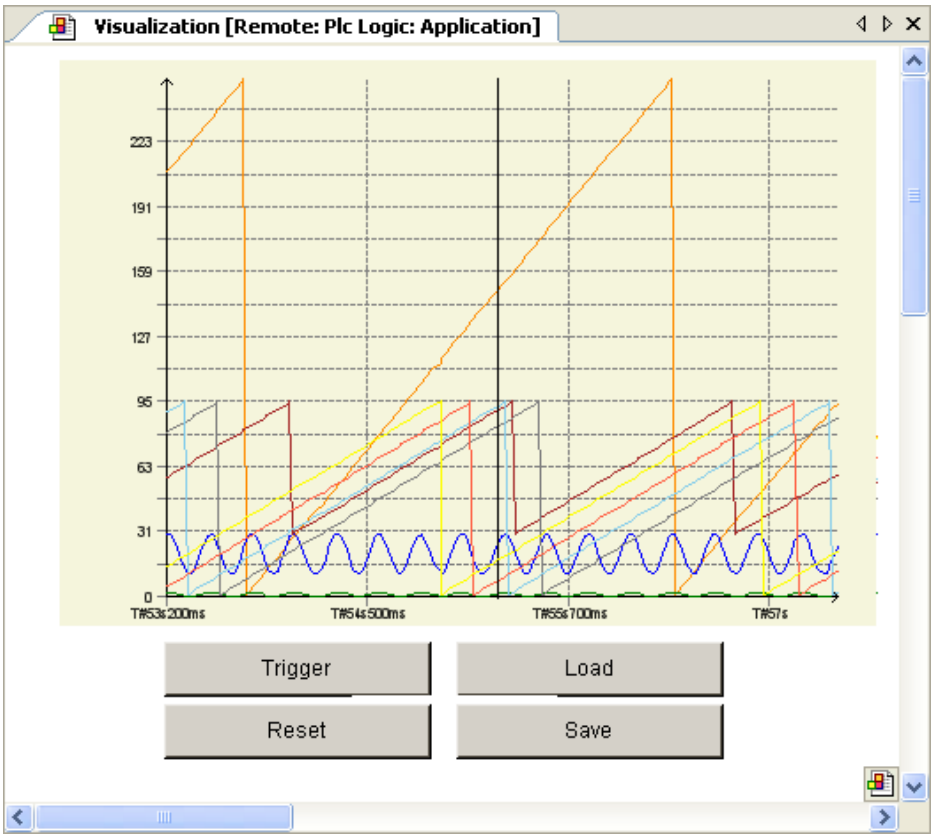


Abb. 65 Beispiel: Trace im Online-Modus

Die Flanke der Trigger-Variablen ist durch eine schwarze vertikale Linie markiert. Die Aufzeichnung der Variablen wird danach bis zu einem angegebenen Prozentsatz (TriggerPostion%) des gesamten Trace-Intervalls fortgeführt, anschließend wird die Visualisierung angehalten.

Bei Aktivierung der Option "ShowCursor" in den Elementeigenschaften des Visualisierungselements Trace werden innerhalb eines Tooltips die Werte der getraceten und der Trigger-Variablen zu der Position angezeigt, die durch den Mauszeiger im Visualisierungselement Trace markiert ist.

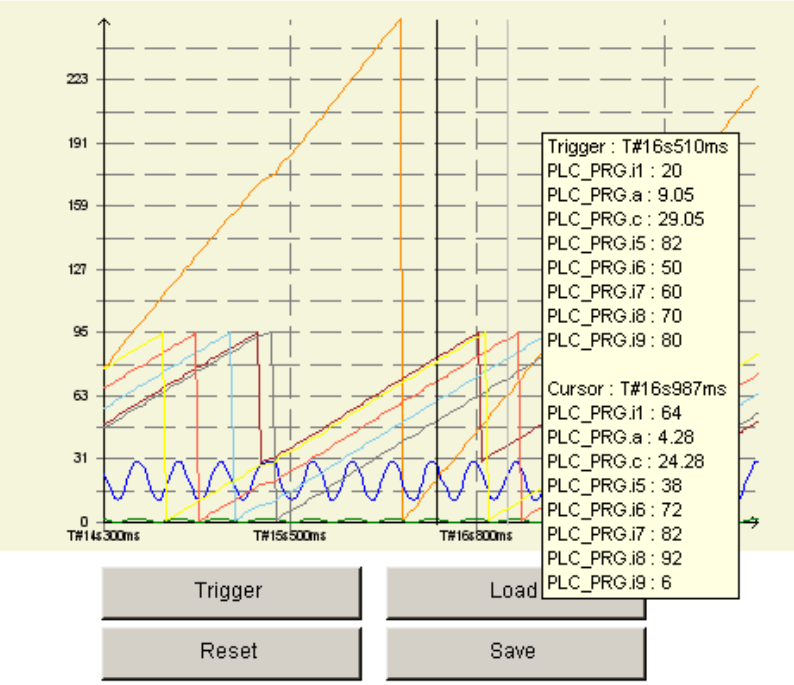


Abb. 66 Beispiel: Effekt der Option "ShowCursor"

Trace-Konfigurationsdatei

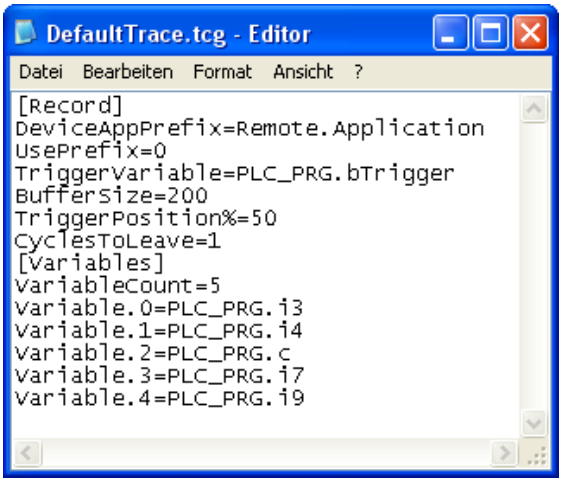


Abb. 67 Beispiel: Trace-Konfigurationsdatei

Die Trace-Konfigurationsdatei <Tracename>_<Recordname>.tcg muss im Verzeichnis des Laufzeitsystems abgelegt werden, beispielsweise in "C:\Programs\ PLC Designer \Gateway PLC".

Abschnitt [Record]: Indem man "DeviceAppPrefix" auf den gemeinsamen Pfadanteil der zu tracenden Variablen setzt, erspart man sich die jeweilige komplette Angabe des Namens bei den einzelnen Trace-Variablen. Dieser Präfix wird auch innerhalb des Tooltips angezeigt, sofern der Parameter "UsePrefix" auf 1 gesetzt ist. Andernfalls, wenn "UsePrefix" auf 0 gesetzt wurde, wird der Präfix innerhalb des Tooltips ausgeblendet.

Abschnitt [Variables]: Die Anzahl der aufgezeichneten Variablen wird in "VariableCount" festgelegt. Dabei kann die Anzahl des ursprünglichen Trace beibehalten oder vermindert werden, nicht jedoch übertroffen. Alle aufzuzeichnenden Variablen müssen über eine Symbol-Konfiguration bekannt gemacht werden; alternativ können sie über eine Datenquelle deklariert werden, welche im den Trace enthaltenden Gerät sitzt.

8.20 Visualisierungselemente vom Typ Windows Controls

Die Kategorie "Windows Controls" des Werkzeuge-Fensters im Visualisierungs-Editor enthält die folgenden Elemente:

Table		Dieses Element dient zur Darstellung der Werte eines Arrays, einer Struktur oder der lokalen Variablen eines Funktionsbausteins. Die Elemente eines eindimensionalen Array bzw. einer Struktur oder einer POU werden mit Hilfe einer Spalte bzw. einer Zeile angezeigt. Zweidimensionale Arrays bzw. Arrays, die als Elemente Strukturen oder POUs beinhalten, werden als Matrixstruktur (Spalten und Zeilen) angezeigt. Eine Variable vom Basistyp kann als eindimensionales Array mit einem Element aufgefasst werden und somit auch mit Hilfe des Table-Elements angezeigt werden (als Tabelle mit einem einzigen Eintrag).
Textfield		Dies Element dient zur Anzeige von Text, der entweder direkt in den Eigenschaften des Elements eingegeben wurde, oder aber von einer Text-Eingabevariable stammt. Im Unterschied zum normalen Rechteck kann der Rahmen schattiert dargestellt werden.
Scrollbar		Diese Element erzeugt eine Bildlaufleiste, deren Minimal- und Maximalwert definiert werden kann. Die Position des Reglers folgt dann dem Wert der Eingabevariablen. Ist andernfalls eine Ausgangsvariable definiert, so kann der Wert dieser durch manuelle Positionierung des Reglers geschrieben werden. Standardmäßig verläuft die Leiste horizontal; wird nach dem Einfügen der Leiste jedoch die Form des Elements (durch Ziehen mit der Maus) so verändert, dass die Höhe die Breite übertrifft, so verwandelt sich die Anzeige in eine vertikale Leiste.

8.20.1 Konfiguration des Elements Table

Ein Table Element kann in eine Visualisierung eingefügt werden um ein- oder zweidimensionale Arrays, Strukturen oder die lokalen Variablen einer POU zu visualisieren.

Die Konfiguration der Tabelle erfolgt innerhalb der Ansicht der **Elementeigenschaften**; detaillierte Information zu den einzelnen Eigenschaften der Visualisierungselemente finden Sie auf dieser Seite der Hilfe. Im folgenden werden einige einführende Beispiele zur Konfiguration von Table Elementen vorgestellt.

Zunächst die durch das Table Element zu visualisierende Variable spezifiziert werden. Hierzu wird im Feld "**Data array**" der Elementeigenschaften ihr kompletter Pfad eingetragen. Die Struktur der Variablen bestimmt die Anzahl an Zeilen und Spalten der Tabelle.

Die **Zeilen** repräsentieren die erste Dimension des angegebenen Arrays. Die **Spalten** repräsentieren im Fall eines zweidimensionalen Arrays den zweiten Index; im Fall einer Struktur repräsentieren sie die einzelnen Komponenten der Struktur, im Fall einer POU die lokalen Variablen. So wird beispielsweise ein eindimensionales Array von Elementen eines einfachen Basistyps mit Hilfe einer einzigen Spalte repräsentiert, wohingegen ein Array von Elementen vom Strukturtyp in einer Tabelle dargestellt wird, deren Anzahl an Zeilen der Anzahl an Array Elementen entspricht und deren Anzahl an Spalten der Anzahl an Strukturkomponenten entspricht.

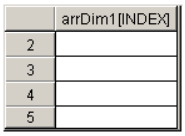
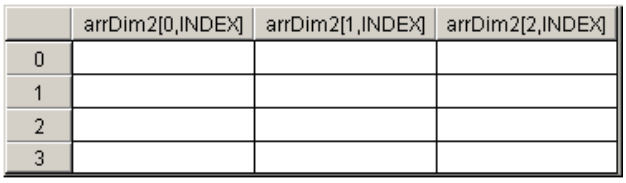
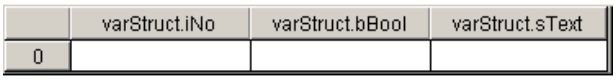
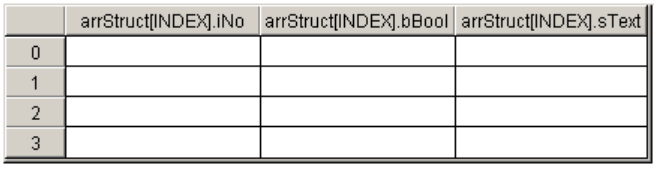
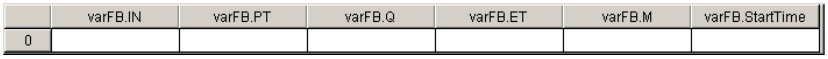
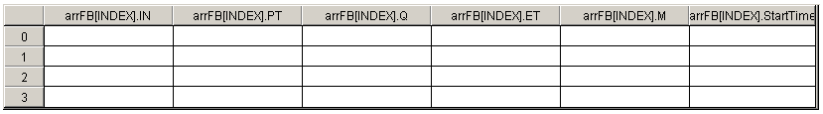
Wird im nachhinein die Dimension der verknüpften Variable verändert, beispielsweise durch Abänderung der Arraygrenzen oder durch Hinzunahme oder Entfernen von Strukturkomponenten, so kann die Tabelle durch Betätigung der **Eingabetaste** (Return) in einem Tabellenfeld **aktualisiert** werden. Dabei bleibt die Konfiguration weiterhin gültiger Komponenten erhalten.

Eine **Tabellenspalte kann dupliziert werden**, indem bei gedrückter <Strg> Taste mit der linken Maustaste in die Titelzelle der zu duplizierenden Spalte geklickt wird.

Eine **Tabellenspalte kann gelöscht werden**, indem bei gedrückten <Strg>+<Umschalt> Tasten mit der linken Maustaste in die Titelzelle der zu duplizierenden Spalte geklickt wird.

Hinweis: Wenn der Eintrag im Feld **Data array** der Elementeigenschaften geändert wird, werden alle übrigen Einstellungen für die Elementeigenschaften wieder auf die Standardeinstellungen zurückgesetzt!

Betrachten Sie folgende Beispiele:

Deklaration der Variablen	**Struktur des zugehörigen Table Elements**
arrDim1 : ARRAY[2..5] OF INT;	 Beachten Sie, dass die Indizierung in der Zeilenbeschriftung mit den Indices der ersten Dimension des Arrays übereinstimmt; infolgedessen startet die Nummerierung der Zeilen in diesem Beispiel mit einer "2"!
arrDim2 : ARRAY[0..2, 0..3] OF INT;	
TYPE MyStruct : STRUCT iNo: INT; bBool : BOOL; sText:STRING; END_STRUCT END_TYPE varStruct: MyStruct;	
arrStruct : ARRAY[0..3] OF MyStruct;	
varFB: TON;	
arrFB : ARRAY[0..3] OF TON;	

Neben einiger grundlegende Eigenschaften wie z. B. Größe, Position oder Linienbreite der Tabelle können eine Vielzahl von spezifischen Eigenschaften für die Tabelle eingestellt werden. Eine Auswahl dieser Eigenschaften soll anhand einer schrittweisen Weiterführung des Beispiels arrStruct näher erläutert werden, ihr Effekt ist jeweils in der rechten Spalte der untenstehenden Tabelle aufgezeigt.

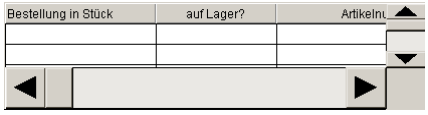
Zunächst ändern wir die standardmäßigen Einträge in der Spaltenbeschriftung in aussagekräftigere Titel ab, passen die Breite der Spalten an und legen die Ausrichtung der Überschrift fest; z.B. für die erste Spalte

Columns	
Column	
[0]	☑
Column header	Bestellung in Stück
Column width	150
Text in headline	LEFT

Es gibt zwei Möglichkeiten, die Breite der Spalten zu ändern: 1.In den Elementeigenschaften: Eintragen der gewünschten Breite im Feld "Column width", oder 2. Direkt im Tabellenelement: Mausklick auf den Trennstrich zwischen zwei Spalten in der Titelzeile und mit der Maus an die gewünschte Position ziehen.

Wir setzen die Höhe der Spalten auf 20 Pixel fest und schalten die Beschriftung der Zeilen aus, so dass nun die erste Spalte, die die Array Indices beinhaltete, fehlt. Das Bild rechts zeigt zudem die verbreiterten Bildlaufleisten ("Scrollbar size"), welche zur Tabelle hinzugefügt werden, sobald diese auf eine Größe unter der Summe der Zeilenhöhen bzw. der Spaltenbreiten geschrumpft wird.

Columns	
Column	
Row header	☐
Column header	☑
Row height	20
Row header width	15
Scrollbar size	40



Um die Bildlaufleisten wieder verschwinden zu lassen passen wir die Größe der Tabelle an: ihre Höhe ergibt sich aus der Zeilenhöhe multipliziert mit der Anzahl an Zeilen (+1 im Fall von eingeschalteter Spaltenbeschriftung); ihre Breite ist die Summe aus den einzelnen Spaltenbreiten und der Breite der Zeilenbeschriftung (sofern diese aktiviert ist). I.a. muss zu diesen Zahlen noch ein kleiner Überhang (von bspw. 1) addiert werden.

⊟ Position	
X	21
Y	221
Width	486
Height	101
⊟ Text properties	
HorizontalAlignment	HCENTER
VerticalAlignment	VCENTER
Font	Tahoma; 12px

Schriftart, -farbe und -größe können innerhalb der Dialogbox ausgewählt werden, welche nach einem Klick auf die Schaltfläche ... im Feld "Font" öffnet. Beachten Sie, dass die Vorgabe für die horizontale Textausrichtung ("HorizontalAlignment")nur für die Zeilenbeschriftung gilt; die Ausrichtung der Texte in der einzelnen Tabellenzellen folgt den hierfür getroffenen Einstellungen in den Spalteneigenschaften.

Die (online) Darstellung der Tabelle zeigt nun ein verändertes Aussehen:

	Bestellung in Stück	auf Lager?	Artikelnummer
0	250	TRUE	DD32F7098367
1	480	FALSE	DD33F1782648
2	1500	TRUE	DD45G2637526
3	2500	TRUE	AD70F1829388

Beachten Sie, dass die Ausrichtung der Einträge in den Tabellenzellen entsprechend der Einstellung für die einzelnen Spalten erfolgt.

Die Tabellenfelder selbst können zeilenweise editiert werden: Aktivieren Sie zu diesem Zweck für die ausgewählte Spalte das Kontrollkästchen "Use template", woraufhin innerhalb der Spalteneigenschaften ein weiterer Punkt "Template" eingefügt und aufgeklappt wird:

Use template	☑
Use text alignment in headline	☐
Template	
Type of element	VISU_ST_RECTANGLE
Colors	
Normalstate	
Framecolor	Coral
Fillcolor	Bisque
Alarmstate	
Framecolor	OrangeRed
Fillcolor	LightSalmon
Elementlook	
Linewidth	1
Fillattributes	BS_SOLID
Frameattributes	PS_DASH
Texts	
Text	%s
Tooltip	
Text properties	
HorizontalAlignment	RIGHT
VerticalAlignment	VCENTER
Font	Copperplate Gothic Light; 12px; style=Bold
Text variables	
Text Variable	PLC_PRG.arrStruct[INDEX].iNo

Drei Typen von Template Elementen stehen zur Auswahl: standardmäßig ist ein Rechteck (rectangle) eingetragen, das jedoch durch Klicken in das entsprechende Feld in ein Rechteck mit abgerundeten Ecken (rounded rectangle) oder einen Kreis (circle) geändert werden kann. Die zugehörigen Konfigurationsmöglichkeiten zeigen die Elementeigenschaften des ausgewählten Template Elements: wie gewöhnlich können Sie die Füll- und Rahmenfarben im Normal- und Alarmzustand wählen, die auch mit Hilfe einer Variablen getoggled werden können. Die Ausrichtung der Einträge in den Zellen der Spalte wird nun (mit Ausnahme der Überschrift) durch die Einstellung im Template bestimmt.

Hinweis: Anstelle der standardmäßig eingetragenen Array-Komponente kann im Feld "Text variables" auch jede andere Variable des Projekts eingetragen werden. Somit ist es insbesondere möglich, die Array-Elemente auch in veränderter Reihenfolge in der Tabelle anzeigen zu lassen.

Ein Vertauschen der Spalten kann auch in der Tabelle selbst vorgenommen werden. Klicken Sie hierzu innerhalb der Titelzeile in die Mitte einer Spalte und ziehen die Spalte mit gedrückter Maustaste auf ihre neue Position.

In untenstehendem Beispiel wurde für die erste und dritte Spalte ein Template benutzt, wobei das Template der ersten Spalte die links aufgezeigten Eigenschaften bestitzt.

	Bestellung in Stück	auf Lager?	Artikelnummer
0	250	TRUE	DD32F7098367
1	480	FALSE	DD33F1782648
2	1500	TRUE	DD45Q2637526
3	2500	TRUE	AD70F1829388

Der Alarmzustand kann nicht nur durch einen Variablenwert ausgelöst werden, sondern auch durch Selektieren einer Tabellenzelle durch einen Mausklick im Online Modus. Das Verhalten der selektierten Region wird im Abschnitt "Selection" festgelegt:

Selection	
Selection color	HotPink
Apply to columns	Apply
Selection type	SEL_ROW
Frame around selected cells	☑
Variable for selection X	PLC_PRG.SelectedPosX
Variable for selection Y	PLC_PRG.SelectedPosY
Variable for valid selection X	PLC_PRG.IsSelectedX
Variable for valid selection Y	PLC_PRG.IsSelectedY

Die hier festgesetzte Hintergrundfarbe für die selektierten Elemente gilt für alle selektierten Zellen, für welche keine Alarmfarbe (innerhalb eines Templates) vorgeschrieben wurde. So wird in unserem Beispiel eine selektierte Zelle der zweiten Spalte in HotPink eingefärbt, wohingegen eine selektierte Zelle der ersten Spalte die Alarm Farbe des zugehörigen Templates (LightSalmon) erhält.
Wenn Sie den Schalter **Apply** drücken, so werden die Alarmfarben der speziellen Templateeinstellungen durch die im Feld "Selection color" getroffenen Auswahl ersetzt.
Folglich erhalten daraufhin die entsprechenden Felder im Template der ersten Spalte unserer Beispiels folgende Einträge:

Template	
Type of element	VISU_ST_RECTANGLE
Colors	
Normalstate	
Framecolor	Coral
Fillcolor	Bisque
Alarmstate	
Framecolor	OrangeRed
Fillcolor	HotPink

Das Feld "Selection type" bestimmt, welche Elemente durch Anklicken einer einzelnen Tabellenzelle selektiert werden sollen: nachdem wir den Typ SEL_ROW gewählt haben, werden alle Zellen, die zur selben Zeile gehören wie die angeklickte Zelle, als selektiert angesehen und umrahmt.

Wurden keine Spalten von der Darstellung in der Tabelle ausgeschlossen, so repräsentiert die Position der selektierten Zelle innerhalb der Tabelle die zum Eintrag gehörenden Indices des visualisierten Arrays. Diese Indices können dann in Variablen gespeichert werden, die hierfür in den letzten drei Zeilen des Abschnitts "Selection" eingetragen werden. Die dritte und vierte Variable sind von booleschem Typ und werden auf TRUE gesetzt, sofern die Selektion gültig ist, was die Zeile bzw. Spalte anbelangt.

Noch einmal das Table Element im Online Modus: Nach Anklicken der Zelle mit dem Eintrag "FALSE" gelten alle Zellen in der zweiten Zeile als selektiert. Nur die selektierte Zelle in der zweiten Spalte trägt die angegebene Auswahlfarbe, nachdem die Farben des Alarmzustands in den anderen Spalten durch die Templates vorgeschrieben werden:

	Bestellung in Stück	auf Lager?	Artikelnummer
0	250	TRUE	DD32F7098367
1	480	FALSE	DD33F1782648
2	1500	TRUE	DD45G2637526
3	2500	TRUE	AD70F1829388

Nach Anwenden der Auswahlfarbe auch für die Templates erhält die Selektion ein verändertes Ausssehen:

	Bestellung in Stück	auf Lager?	Artikelnummer
0	250	TRUE	DD32F7098367
1	480	FALSE	DD33F1782648
2	1500	TRUE	DD45G2637526
3	2500	TRUE	AD70F1829388

Die verknüpften Variablen erhalten als Wert den zur ausgewählten Zelle gehörenden Array Index:

CoDeSys_SP_Win_V3.Application.PLC_PRG

Ausdruck	Datentyp	Wert
arrDim1	ARRAY [2..5] OF INT	
arrDim2	ARRAY [0..2, 0..3] O...	
varStruct	MyStruct	
arrStruct	ARRAY [0..3] OF My...	
varFB	TON	
arrFB	ARRAY [0..3] OF TON	
iCounter	INT	277
iRow	INT	0
iCol	INT	7
SelectedPosX	INT	2
SelectedPosY	INT	1
IsSelected	BOOL	TRUE

8.21 Tastaturbedienung im Online-Modus

Für die Tastaturbedienung einer Visualisierung Im Online-Modus werden einige Standard-Tastaturkürzel von jedem Gerät unterstützt. Zusätzlich können geräteabhängig weitere Tasten(kombinationen) anwendbar sein.

Standard-Tastaturkürzel:

Taste(n)	Aktion
Tab	Nächstes Element gemäß der chronologischen Reihenfolge der Einfügung wird ausgewählt; innerhalb einer Tabelle wird dabei jede einzelne Zelle angewählt; wenn ein Frame-Element ausgewählt ist, wird die Selektion danach an die einzelnen enthaltenen Elementen weitergegeben
Shift + Tab	Vorheriges Element wird ausgewählt; umgekehrte Reihenfolge wie mit Tab
Enter	Eingabe-Aktion auf das ausgewählte Element wird ausgeführt
Arrow keys	Nächstes Element in der durch die Pfeiltaste gegebene Richtung wird ausgewählt

Hinweis: Für die integrierte Visualisierung kann die Tastaturbedienung über den Befehl Tastaturbedienung aktivieren explizit aktiviert und deaktiviert werden. Dies kann erwünscht sein, weil solange die Tastaturbedienung für die Visualisierung aktiviert ist, andere Befehle, die über Tastaturkürzel gegeben werden, nicht ausgeführt werden.

Gerätespezifische Tastaturbedienung:

Neben den Standard-Tasten hängt es vom Gerät ab, welche weiteren Tasten(kombinationen) für die Visualisierungen, die auf dem Gerät laufen, verwendet werden können. Siehe dazu Standard Tastaturkürzel im Visualization Manager bzw. Tastaturkonfiguration im Visualisierungs-Editor.

Hinweis: Es gibt die Möglichkeit, Tastaturereignisse im Applikations-Code zu behandeln.

8.22 Events und Eingabeaktionen bemerken

Mit Hilfe von Methoden, die von den Visualisierungsbibliotheken bereitgestellt werden, kann die Applikation bestimmte Ereignisse (Events), die durch Benutzereingaben in einer Visualisierung ausgelöst werden, wahrnehmen:

- Schließen von Edit Controls
- Tastatur-Ereignisse

Schließen eines Edit Controls (Schreiben einer Variable) bemerken

Die Applikation kann feststellen, wann ein "Edit Control" in einer zugehörigen Visualisierung geschlossen wird, d.h. wenn ein Variablenwert durch eine Benutzereingabe in ein Eingabefeld geschrieben wird. Dies kann nützlich sein, wenn eine weitere Aktion nach dem Schließen eines Edit Controls ausgeführt werden soll.

Um über das Schließen informiert zu werden, kann eine Funktionsblock-Instanz (die die Schnittstelle VisuElems.IEditBoxInputHandler aus VisuElemBase.library implementiert) der globalen Instanz VisuElems.Visu_Globals.g_VisuEventManager zugewiesen werden, wobei die Methode SetEditBoxEventHandler dieser Instanz verwendet wird, wie im Beispiel unten gezeigt.

VisuElems.IEditBoxInputHandler benötigt die Methode VariableWritten, die aufgerufen wird, nachdem durch eine Benutzereingabe in einer Visualisierung ein Variablenwert geschrieben wurde.

Beispiel:

Es gibt eine Visualisierung mit den folgenden Elementen:

1. Zwei Rechtecke, jedes mit einem Edit Control Feld (Eingabe "Variable schreiben"), in das der Benutzer einen Wert für Variable i oder j eingeben kann.

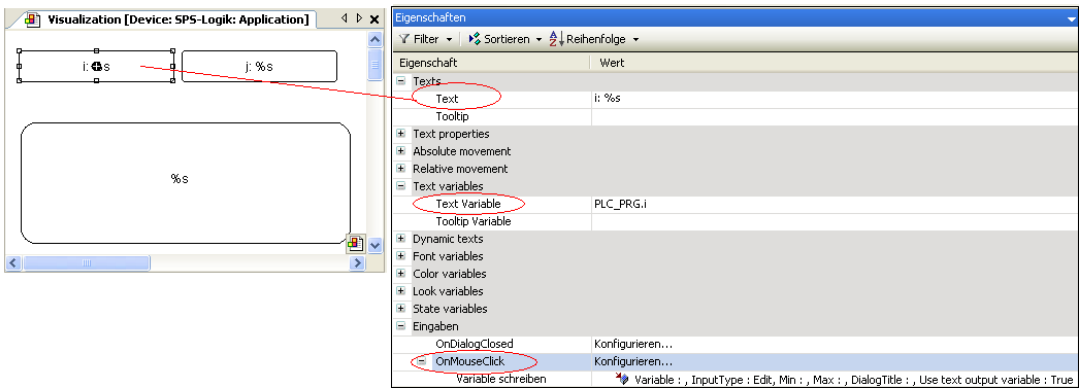


Abb. 68 Eingabe-Elemente

2. Ein weiteres Rechteck, in dem ein Text (Text Variable stInfo) mit einigen Informationen angezeigt wird, wenn eines der Eingabefelder in den oberen beiden Rechtecken geschlossen wird, d.h. wenn ein Wert geschrieben wird.

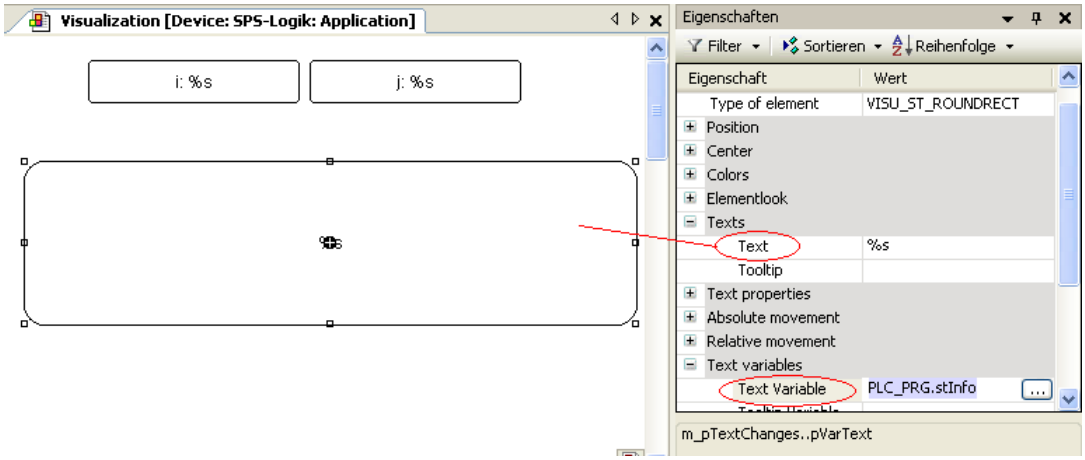
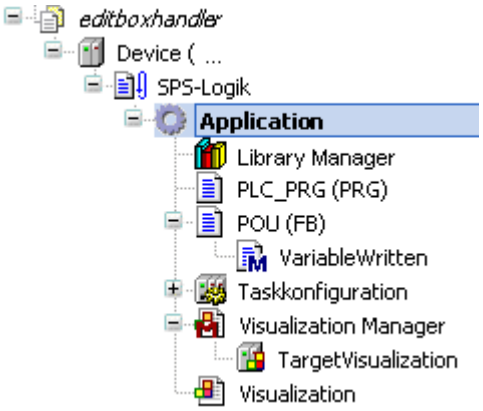


Abb. 69 Textausgabe-Element

Die Applikation enthält die folgenden Bausteine:



Programm PLC_PRG:

```
PROGRAM PLC_PRG
VAR_INPUT
        i:INT;  (* Variable, deren Wert der Benutzer über die Visualisierung
        schreiben kann *)
        j:REAL; (* Variable, deren Wert der Benutzer über die Visualisierung
        schreiben kann *)
        stInfo : STRING;  (* Information zur Benutzereingabe über das Edit
        Control Feld; Text wird über die Methode VariableWritten
        zusammengesetzt; Ergebnis wird im unteren Rechteck der
        Visualisierung dargestellt *)
END_VAR
VAR
        inst : POU;
        bFirst : BOOL := TRUE;
END_VAR

IF bFirst THEN
  bFirst := FALSE;
  VisuElems.Visu_Globals.g_VisuEventManager.SetEditBoxEventHandler(inst); (*
  Aufruf der Methode VariableWritten *)
END_IF
```

Funktionsblock POU:

```
FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IEditBoxInputHandler
(keine weiteren Deklarationen, kein Implementierungs-Code)
```

Methode VariableWritten, gehört zu POU:

```
METHOD VariableWritten : BOOL   (* gibt Informationen, wenn ein Edit Control in der Visualisierung geschlossen wird, d.h. wenn eine Variable durch eine Benutzereingabe in einem der oberen Rechtecke geschrieben wird *)
VAR_INPUT
        pVar : POINTER TO BYTE;
        varType : VisuElems.Visu_Types;
        iMaxSize : INT;
        pClient : POINTER TO VisuElems.VisuStructClientData;
END_VAR

// String stinfo, der im unteren Rechteck dargestellt werden soll, wird hier zusammengesetzt
PLC_PRG.stInfo := 'Variable written; type: ';
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, INT_TO_STRING(varType));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', adr: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(pVar));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', by: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, SEL(pClient^.globaldata.clienttype = VisuElems.Visu_ClientType.Targetvisualization, 'other visu', 'targetvisu'));
```

Tastatur-Ereignisse bemerken

Die Applikation kann Tastenaktionen (Tastatur-Event) in einer zugehörigen Visualisierung feststellen, d.h. sie bemerkt, wenn der Anwender eine **Taste drückt und wieder losläßt** wenn die Visualisierung aktiv ist.

Hinweis: Beachten Sie die Konfigurationsmöglichkeiten für die Tastaturbedienung von Visualisierungen.

Um über solche Events informiert zu werden, kann ein Funktionsbaustein (der die Schnittstelle VisuElems.IVisuUserEventManager aus VisuElemBase.library implementiert) der globalen Instanz VisuElems.Visu_Globals.g_VisuEventManager zugewiesen werden und die Methode **SetKeyEventHandler** dieser Instanz wie im folgenden Beispiel verwendet werden:

Beispiel:

Es gibt eine Visualisierung mit einem Visualisierungselement, das den Wert PLC_PRG.stInfo als Text anzeigt.

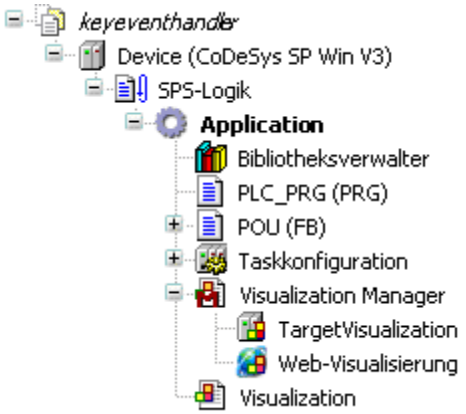
Das heißt, im Online-Betrieb wird in diesem Element zur gerade ausgelösten Tastenaktion die Information "KeyEvent up: <dwKey>, key: <key id>, modifier: <dwModifiers>, by: <pClient^.globaldata.clienttype.....>" ausgegeben.

Beispiel: "KeyEvent up: TRUE, key: 75, modifier: 0, by: targetvisu".

Eigenschaft	Wert
Elementname	GenElemInst_2
Type of element	VISU_ST_ROUNDRECT
⊞ Position	
⊞ Center	
⊞ Colors	
⊞ Elementlook	
⊟ Texts	
Text	%s
Tooltip	
⊞ Text properties	
⊞ Absolute movement	
⊞ Relative movement	
⊟ Text variables	
Text Variable	PLC_PRG.stInfo
Tooltip Variable	
⊞ Dynamic texts	

Abb. 70 Textkonfiguration des Visualisierungselements

Die Applikation enthält die folgenden Bausteine:



Programm PLC_PRG:

```
PROGRAM PLC_PRG
VAR_INPUT
        stInfo : STRING;
END_VAR
VAR
        inst : POU;
        bFirst : BOOL := TRUE;
END_VAR

IF bFirst THEN
  bFirst := FALSE;
  VisuElems.Visu_Globals.g_VisuEventManager.SetKeyEventHandler(inst);
END_IF
```

Dies ermöglicht festzustellen wenn ein Tastatur-Event ausgelöst wird.

Funktionsblock POU:

```
FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IKeyEventHandler
(keine weiteren Deklarationen, kein Implementierungs-Code)
```

Methode HandleKeyEvent, POU zugewiesen:

Die Schnittstelle VisuElems.IVisuUserEventManager benötigt eine Methode mit folgender Signatur, die mit aller verfügbaren Information aufgerufen wird:

```
/// Diese Methode wird aufgerufen, nachdem ein Tastatur-Event ausgelöst wurde.
/// RETURN:
/// TRUE - Wenn der EventHandler dieses Ereignis bearbeitet hat und es nicht von nicht von anderer Stelle behandelt werden sollte
/// FALSE - Wenn das Ereignis nicht von diesem Handler bearbeitet wird
METHOD HandleKeyEvent : BOOL
VAR_INPUT
/// Event-Typ. Der Wert ist TRUE, wenn ein key-up Ereignis stattfand.
bKeyUpEvent : BOOL;
/// Key code
dwKey : DWORD;
/// Modifier. Mögliche Werte:
/// VISU_KEYMOD_SHIFT : DWORD := 1;
/// VISU_KEYMOD_ALT : DWORD := 2;
/// VISU_KEYMOD_CTRL : DWORD := 4;
dwModifiers : DWORD;
/// Zeiger auf die Struktur des Clients wo das Event ausgelöst wurde
pClient : POINTER TO VisuStructClientData;
END_VAR
VAR
END_VAR

PLC_PRG.stInfo := 'KeyEvent up: ';
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, BOOL_TO_STRING(bKeyUpEvent));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', key: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(dwKey));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', modifier: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(dwModifiers));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', by: ');
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, SEL(pClient^.globaldata.clienttype = VisuElems.Visu_ClientType.Targetvisualization, 'other visu', 'targetvisu'));
```

8.23 Eingabe-Dialoge

Eine Visualisierung kann zum Zweck der späteren Verwendung als Eingabe-Dialog entsprechend erstellt und in ihren Objekteigenschaften als "Dialog" defniert werden. Dialog-Visualisierungen können dann in anderen Visualisierungen verwendet werden, um dem Benutzer eine Eingabemaske bereitzustellen. Dialog-Visualisierungen werden beispielsweise bei der Konfiguration einer "OpenDialog" oder "OnDialogClosed" Eigenschaft eines Visualisierungselements angeboten.

Neben selbst-erstellten Dialog-Visualisierungen kann die Standard-Login-Dialog-Visualisierung verwendet werden, wenn die Bibliothek **VisuDialogs.library** im Projekt eingebunden ist.

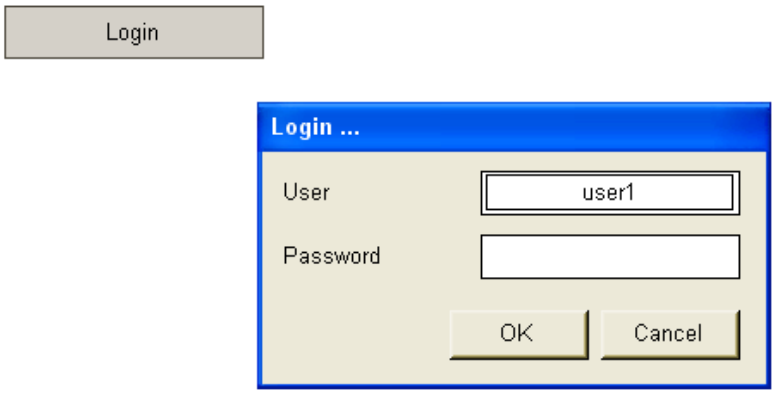
Die in der **Schnittstelle** (Schnittstellen Editor) einer Dialog-Visualisierung definierten Parameter werden automatisch in eine Struktur **<Dialogname>_VISU_STRUCT**. ("Dialogname" = Name der Visualisierung) geschrieben. Diese Struktur kann dann von der Applikation verwendet werden, z.B. über eine Funktion, deren Aufruf im Falle einer Benutzereingabe auf ein Visualisierungselement erfolgt (Konfiguration des Visualisierungselements). So kann das Öffnen eines Dialogs in einer Visualisierung und die Reaktion auf Benutzereingaben in diesem Dialog programmiert werden.

DialogManager

Alle Visualisierungen, die als "Dialog" konfiguriert sind, werden automatisch instanziert und vom internen **DialogManager** verwaltet. Dieser Manager kann über den ebenfalls intern arbeitenden **VisuManager** angesprochen werden, indem die Methode **GetDialogManager()** aufgerufen wird. Der DialogManager stellt einige Methoden bereit, über die die Applikation einen Visualisierungsdialog handhaben kann.

Beispiel für die Verwendung des Login-Dialogs aus der Bibliothek VisuDialog.library:

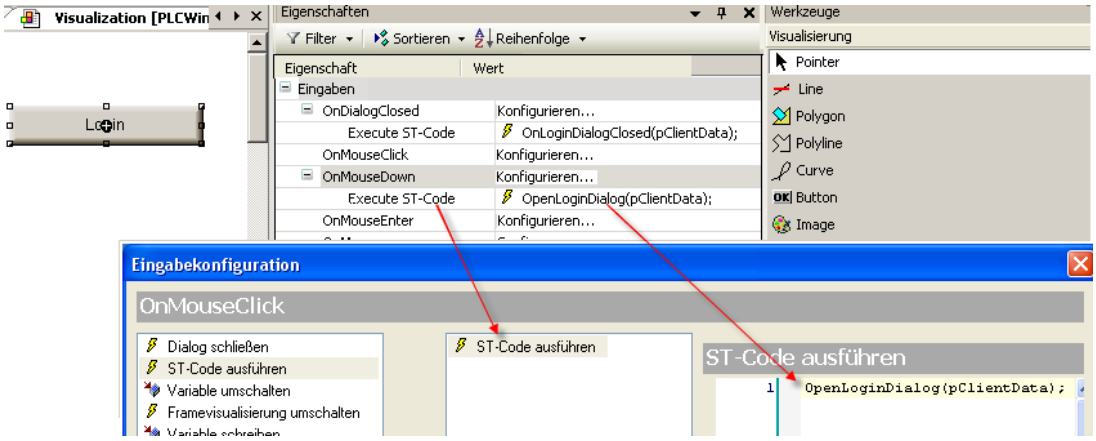
In diesem Beispiel wird ein Schaltflächenelement in einer Visualisierung so konfiguriert, dass bei einem Mausklick auf die Schaltfläche ein Login-Dialog geöffnet wird, in dem ein Benutzername und ein Passwort eingegeben werden können. Für diesen Fall wird ein Standard-Login-Dialog verwendet, der von der VisuDialog.library bereitgestellt wird. Aber auch ein selbsterstellter Dialog könnte in gleicher Weise gehandhabt werden:



Konfiguration der "Login" Schaltfläche:

Ein Schaltflächenelement wird in eine Visualisierung eingefügt. Es wird mit der Text-Eigenschaft "Login" und den folgenden zwei Eingabe-Eigenschaften konfiguriert:

1. OnMouseDown / ST-Code ausführen: Funktionsaufruf OpenLoginDialog(pClientData). Variable pClientData muss an eine entsprechende Funktion OpenLoginDialog übergeben werden, die im Projekt vorhanden ist.



2. OnDialogClosed / Dialog schließen / ST-Code ausführen: Funktionsaufruf OnLoginDialogClosed(pClientData)

Die entsprechenden Funktionen OpenLoginDialog und OnLoginDialogClosed sind als Bausteine im Projekt vorhanden. Siehe im folgenden eine mögliche Implementierung:

Implementatierung der Funktionen OpenLoginDialog und OnLoginDialogClosed:

1. Funktion OpenLoginDialog, wird aufgerufen wenn der Dialog geöffnet wird, z.B. bei einer OnMouseDown-Aktion auf das Schaltflächenelement.

Diese Funktion liest die Parameter des geöffneten Dialogs.

```
FUNCTION OpenLoginDialog : bool
VAR_INPUT
  pClientData : POINTER TO VisuStructClientData;
END_VAR
VAR
  dialogMan : IDialogManager;
  loginDialog : IVisualisationDialog;
  pLoginInfo : POINTER TO Login_VISU_STRUCT; // Login_VISU_STRUCT ejnthält
  die Parameter, die in der Schnittstelle der Visualisierung "Login" definiert sind.
  result : Visu_DialogResult;
  stTitle : STRING := 'Login ...';
  stPasswordLabelText: STRING;
  stUserLabelText: STRING;
  stUsername: STRING;
END_VAR
dialogMan := g_VisuManager.GetDialogManager();    // Der DialogManager wird
durch den implizit verfügbaren VisuManager bereitgestellt
IF dialogMan <> 0 AND pClientData <> 0 THEN
        loginDialog := dialogMan.GetDialog('VisuDialogs.Login'); // Der zu
        öffnende Dialog wird angegeben
        IF loginDialog <> 0 THEN
                pLoginInfo := dialogMan.GetClientInterface(loginDialog,
                pClientData);
                IF pLoginInfo <> 0 THEN     // im folgenden werden die
                Parameter des Login-Dialogs aus Login_VISU_STRUCT gelesen
                        pLoginInfo^.stTitle := stTitle;
                        pLoginInfo^.stPasswordLabelTxt :=
                        stPasswordLabelText;
                        pLoginInfo^.stUserLabelTxt := stUserLabelText;
                        dialogMan.OpenDialog(loginDialog, pClientData, TRUE,
                        0);
                END_IF
        END_IF
END_IF
```

2. Funktion OnLoginDialogClosed, definiert die Reaktion auf ein Schließen des Dialogs:

```
FUNCTION OnLoginDialogClosed : bool
VAR_INPUT
  pClientData : POINTER TO VisuStructClientData;
END_VAR
VAR
  dialogMan : IDialogManager;
  loginDialog : IVisualisationDialog;
  pLoginInfo : POINTER TO Login_VISU_STRUCT;
  result : Visu_DialogResult;
  stPassword: STRING;
  stUsername: STRING;
END_VAR

dialogMan := g_VisuManager.GetDialogManager();     // Der DialogManager wird vom implizit verfügbaren VisuMAnager bereitgestellt
IF dialogMan <> 0 AND pVisuClient <> 0 THEN
        loginDialog := dialogMan.GetDialog('VisuDialogs.Login'); // erkennt den Login-Dialog
        IF loginDialog <> 0 THEN
                result := loginDialog.GetResult(); // erkennt das Ergebnis (OK, Cancel) des Dialogs
                IF result = Visu_DialogResult.OK THEN
                        loginDialog.SetResult(Visu_DialogResult.None); // Zurücksetzen auf Default (none)
                        pLoginInfo := dialogMan.GetClientInterface(loginDialog, pVisuClient); // Struktur Login_VISU_STRUCT wird gelesen; im folgenden können die Strukturparameter gesetzt werden
                        IF pLoginInfo <> 0 THEN
                                stPassword := pLoginInfo^.stPassword;
                                pLoginInfo^.stPassword := ''; // Passwort wird zurückgesetzt
                                stUsername := pLoginInfo^.stUsername;
                        END_IF
```

```
                        ELSIF result = Visu_DialogResult.Cancel THEN
                                loginDialog.SetResult(Visu_DialogResult.None);
                                // Reaktion auf 'Abbrechen'
                        ELSE
                                // nichts zu tun
                        END_IF
        END_IF
END_IF
```

9 Symbolkonfiguration

Die Symbolkonfiguration wird verwendet, um **Symbole** mit bestimmten Zugriffsrechten zu erzeugen, über die von extern auf Projekt(applikations)variablen zugegriffen werden kann, z.B. von einem OPC-Server aus. Eine Beschreibung der Symbole (Symbol Information) wird in einer xml-Datei (Symboldatei) des Projektverzeichnisses zur Verfügung gestellt und zeitgleich mit der Applikation auf die Steuerung geladen.

Dazu wird eine "**Symbolliste**" angelegt, die zum einen in eine XML-Datei im Projektverzeichnis exportiert wird und zum anderen während des Applikations-Downloads in einer für den Benutzer nicht sichtbaren Datei auf das Zielsystem geladen wird.

Die XML-Datei wird nach folgender Syntax benannt: <Project name>.<Device name>.Application name>.xml

Anlegen einer Symbolkonfiguration:

Im Projekt muss ein Objekt "**Symbolkonfiguration**" unterhalb der betreffenden Applikation im Geräte-Fenster eingehängt werden (Objekt hinzufügen), womit automatisch die Bibliothek **IECVarAccess.library** in den Bibliotheksmanager eingefügt wird..

Die Variablen, die als Symbole exportiert werden sollen, können entweder im Symbolkonfigurations-Editor definiert werden oder aber mit Hilfe von Pragmas, welche den Deklarationen der Variablen hinzugefügt werden. Eine weitere Möglichkeit stellt beispielsweise der AS-Editor bereit: Hier können die in den Element Eigenschaften implizit erzeugten Elementflags zum Export in die Symbolkonfiguration bestimmt werden.

Der **Name eines Symbols** wird in der Symbolkonfiguration gemäß folgender Syntax erzeugt:

<Applikationsname>.<POU-Name>.<Variablenname>

z.B. "MyApplication.PLC_PRG.a" oder "MyApplication.SymFB.a"

Beim Zugriff auf die Variable muss der Symbolname immer vollständig gemäß dieser Schreibweise angegeben werden.

Symbol Information:

Die für eine Applikation definierten Symbole werden in eine xml-Datei innerhalb des Projektverzeichnisses exportiert (**symbol file**). Diese Datei ist laut folgender Syntax benannt:

<project name>.<device name>.<application name>.xml,

beispielsweise also proj_xy.PLC1.application.xml.

Hinweis: Die Symboldatei wird angelegt, wenn im "Erstellen" Menü der Befehl "Code erzeugen" ausgeführt wird.

Die Information wird zusammen mit der Applikation auf die Steuerung **geladen**. Abhängig von der Gerätebeschreibung ist sie direkt in der Applikation enthalten oder wird eine zusätzliche (Kind-)Applikation erzeugt, welche hernach ebenso im Dialog Applikationen des PLC Geräteeditors aufgeführt wird. Die Syntax der Symbol-Applikation lautet:

<application name>._Symbols

Wurde die Symbolkonfiguration beispielsweise für die Applikation "Application"definiert, so wird beim Laden der Applikation gleichzeitig die Symbol Information als "**<Applikationsname>.__Symbols>**" auf die Steuerung geladen:

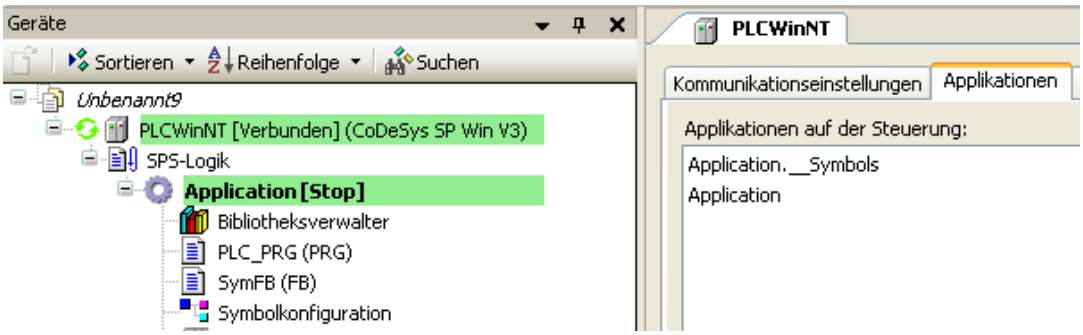


Abb. 71 Beispiel, Applikation und "Symbol Applikation"

Wenn die Symbolkonfiguration im Online-Status geändert wird, kann sie explizit neu geladen werden, siehe Schaltfläche Download im Editorfenster.

Die Symbol-Applikation wird z. B. hinsichtlich der maximalen Anzahl an Applikationen auf einer Steuerung als "normale" Applikation betrachtet.

10 Quellcodeverwaltung

10.1 Quellcodeverwaltung über ENI

Allgemein

Die Objekte eines PLC Designer-Projekts können über ENI (Engineering Interface) in einer **externen Datenbank** verwaltet werden. Die Verwendung einer solchen Datenbasis garantiert die Konsistenz der Daten, die dann von mehreren Anwendern, Projekten und Applikationsprogrammen gleichzeitig genutzt werden können. Außerdem erweitert es die Funktionalität von PLC Designer um folgende Punkte:

- **Versionsverwaltung** für Projektobjekte: Wenn ein Objekt aus der Datenbank ausgecheckt, verändert und wieder eingecheckt wurde, wird in der Datenbank eine neue Version des Objekts angelegt. Aber auch die bisherigen Versionen werden gespeichert und können bei Bedarf wieder abgerufen werden. Für jedes Objekt und für das gesamte Projekt wird die Versionshistorie aufgezeichnet. Zwei Versionen können auf Unterschiede hin geprüft werden.
- **Mehrbenutzerbetrieb**: Die neueste Version einer Bausteinsammlung, z.B. der Bausteine eines Projekts, kann einer Gruppe von Anwendern zugänglich gemacht werden. Die von einem Benutzer ausgecheckten Bausteine sind für die anderen Benutzer als 'in Bearbeitung' markiert und nicht veränderbar. Somit können mehrere Anwender parallel am gleichen Projekt arbeiten, ohne gegenseitig Objektversionen zu überschreiben.
- **Zugriff durch externe Programme**: Neben dem PLC Designer-Programmiersystem können auch andere Tools, die ebenfalls über die ENI-Schnittstelle verfügen, auf die gemeinsame Datenbank zugreifen. Beispielsweise externe Visualisierungen, ECAD-Systeme etc., die die in »PLC Designer« erzeugten Daten benötigen oder selbst Daten erzeugen.

ENI (Engineering Interface)

Das ENI ist eine Schnittstellenkomponente, die es ermöglicht, das PLC Designer-Entwicklungssystem an eine externe Datenbank anzuknüpfen.

Das ENI setzt sich zusammen aus einem **Client**- und einem **Server**-Teil. Dadurch ist es möglich, die Datenbank auf einem entfernten Rechner zu halten, der für Mehrbenutzerbetrieb benötigt wird. Das »PLC Designer« Programmiersystem ist ebenso ein Client des eigenständigen ENI Server Prozesses wie gegebenenfalls eine andere Applikation, die Zugang zur Datenbank braucht.

ENI unterstützt Datenbanksysteme wie z.B. 'Visual SourceSafe 6.0', und ein lokales Dateisystem. Sehen Sie für detaillierte Informationen zum ENI Server bitte die **separate ENI Server Dokumentation**. Objekte können an verschiedenen Orten der Datenbank abgelegt werden. Sie können ausgecheckt und bearbeitet werden und sind in dieser Zeit für andere Benutzer gesperrt. Jeweils die letzte Version eines Objekts wird durch einfaches Abrufen aus der Datenbank erhalten. Parallel können Objekte auch nur lokal im Projekt verwaltet werden, wie es für Projekte ohne Quellcodeverwaltung der Fall ist.

Arbeiten mit einer ENI Projektdatenbank

Beachten Sie bitte die Unterschiede zu PLC Designer V2.3.

Hinweis: Für eine Anleitung zu Installation und Gebrauch des von Lenze bereitgestellten *ENI Servers* sehen Sie bitte die separate Server-Dokumentation bzw. Online Hilfe. Dort finden Sie auch ein "Quickstart"-Kapitel.

Voraussetzungen:

- die Kommunikation zwischen PLC Designer und einem ENI-Server erfordert **TCP/IP**, da der ENI-Server das HTTP-Protokoll verwendet.
- ein **ENI Server** (ENI Server Suite) muss auf dem lokalen System oder auf einem entfernten Rechner installiert und gestartet sein. Zur Verwendung mit einem der Standard-Datenbanktreiber ist eine Lizenz erforderlich, nur der Treiber für ein lokales Dateisystem kann lizenzfrei verwendet werden.
- im ENI-Server Service-Kontroll-Programm (ENI Control) muss die **Verbindung zur gewünschten Datenbank** (siehe nächster Punkt) korrekt konfiguriert sein (Sektion 'Data base'). Sie werden dazu automatisch während der Installation aufgefordert, können die Einstellungen aber auch später in ENI Control wieder ändern.
- eine **Projektdatenbank**, für die ein ENI-unterstützter Treiber verfügbar ist, muss installiert sein. Es wird empfohlen, dies auf dem gleichen Rechner zu tun, auf dem auch der ENI-Server läuft. Alternativ kann ein lokales Dateisystem verwendet werden, für das der entsprechende Treiber standardmäßig mit installiert wird.
- in der Datenbank-Administration müssen gegebenenfalls sowohl der Anwender (am Client) als auch der ENI Server als **Benutzer mit Zugangsrechten** registriert sein. Dies gilt in jedem Fall für die Verwendung von SourceSafe als Datenbank, für andere Datenbanktreiber sehen Sie bitte die zugehörige Dokumentation für die erforderliche Benutzerkonfiguration.
- für das aktuelle PLC Designer **Projekt** muss die **ENI-Schnittstelle aktiviert** sein (dies erfolgt im PLC Designer-Dialog 'Projekt' 'Quellcodeverwaltung' 'Quellcodeverwaltung ändern...'.
- der Benutzer muss mit Name und Passwort beim ENI-Server angemeldet sein; dies erfolgt über den Anmelden-Dialog, der explizit über Befehl Projekt' 'Quellcodeverwaltung' 'Anmelden...' geöffnet werden kann, bzw. der automatisch erscheint, wenn man einen Zugriff auf die Datenbank versucht, ohne vorher im ENI-Server eingeloggt zu haben. Beachten Sie, dass im Unterschied zu PLC Designer V2.3 der Benutzer sich nur einmal - und nicht für jedes Projekt separat - beim ENI-Server anmelden muss, um in verschiedenen Projekten in der Datenbank zu arbeiten.

Arbeiten mit PLC Designer-Projektobjekten unter Quellcodeverwaltung :

- Die **'Quellcodeverwaltungs**'-Befehle (Letzte Version abrufen, Auschecken, Einchecken, Versionsgeschichte, Bezeichnung ect.), die für das Arbeiten mit Objekten in der ENI Projektdatenbank gebraucht werden, sind im PLC Designer-Projekt verfügbar, sobald die Datenbank korrekt aktiviert und konfiguriert wurde.
- Die aktuelle Zuweisung eines Objekts zu einem Datenbank-Speicherort ("**Verknüpfung**") wird in den Objekt-Eigenschaften, Kategorie 'Quellcodeverwaltung' angezeigt und kann dort verändert werden. Eine Übersicht über alle Objekt-Speicherorte in der Datenbank und ebenfalls die Möglichkeit zur Änderung erhält man in den 'Erweiterten Einstellungen' im Dialog 'Quellcodeverwaltung ändern...'.
- Die **allgemeinen Einstellungen** für Quellcodeverwaltungs-Aktionen, z.B. ob immer die neueste Version eines Objekts abgerufen werden soll, bevor ein Projekt übersetzt wird, können im Dialog Optionen / Quellcodeverwaltung vorgenommen werden.

10.2 Unterschiede zu PLC Designer V2.3

Die Quellcodeverwaltung über ENI ist im Vergleich zu PLC Designer V2.3 vereinfacht:

- **Keine Objektkategorien, sondern "Verknüpfungen" von Objekten zu Datenbankorten**:
  Die Kategorien "Gemeinsame Objekte" und "Übersetzungsdateien" gibt es nicht mehr. Alle Objekte sind quasi "Projekt"Objekte, können aber mit beliebigen Datenbankorten verknüpft sein, die sich vom Standard-Datenbankverzeichnis für die Projektobjekte unterscheiden. Objekte, auf die auch andere ENI-Datenbank-Clients zugreifen können sollen (bisher als "Gemeinsame Objekte" angelegt), müssen nicht mehr in einem bestimmten "gemeinsamen" Datenbankverzeichnis liegen, sondern neben dem Default-Projekt-Datenbankverzeichnis können einer oder mehrere Datenbankorte für diesen Zweck definiert sein.
- **Verbindung zum ENI Server muss nur einmal definiert werden**:
  Die Konfiguration der Datenbankverbindung (ENI Server) zum Standard-Datenbankverzeichnis ist nur mehr in einem einzigen Dialog zu definieren (in PLC Designer V2.3 musste dies in drei verschiedenen Dialogen, für Projektobjekte, GEmeinsame Objekte und Übersetzungsdateien gemacht werden). Nur wenn Verknüpfungen einzelner Projektobjekte zu anderen Datenbankorten gewünscht sind, müssen diese explizit angelegt werden. Dies ist jedoch in einem einzigen Dialog für alle Objekte möglich.
- **Die Baumstruktur der Projektobjekte in der Datenbank entspricht immer derjenigen im Projekt**:
  Die Projektobjekte werden immer im Standard-Datenbankort in einer Baumstruktur abgelegt, die derjenigen im POUs- und Gerätefenster in PLC Designer entspricht.

- **Authentifizierung für den Datenbankzugang muss nur einmal erfolgen, für den jeweiligen ENI Server**:
  Die Authentifizierung für den Datenbankzugang ist nicht mehr an ein Projekt oder gar Objektkategorien gebunden, sondern an den ENI Server. Wenn einmal in den entsprechenden ENI Server eingeloggt wurde und die Option zum Speichern der Zugangsdaten aktiviert war, bleibt die Authentifizierung gültig bis der Login-Dialog erneut verwendet wird, auch wenn zwischenzeitlich PLC Designer neu gestartet wurde. Wenn die Zugangsdaten nicht gespeichert wurden, muss bei einem Neustart von PLC Designer eingeloggt werden.

11 Projekt-Dokumentation

Eine Dokumentation d.h. eine druckbare Version des Projekts kann mit den durch das 'PrintEngine'-Plug-In bereitgestellten Funktionen erstellt werden.

Ausdrucke des gesamten Projekts oder einzelner Objekte, können - wenn gewünscht mit Titelseite und Inhaltsverzeichnis - können erzeugt werden.

Befehle der Kategorie 'Drucken' stehen bereit, um in entsprechenden Dialogen den Umfang, das Seitenlayout und die Druckereinetsllungen für die Projekt-Dokumentation zu definieren.

12 Menübefehle, nach Kategorien

12.1 Befehle und Befehlskategorien

Welche Befehle in der »PLC Designer« Benutzeroberfläche verfügbar sind, hängt vom aktuellen Profil ab, d.h. von den aktuell installierten Plug-In Komponenten.

Wo - in welchen Menüs - die Befehle verfügbar sind, hängt grundsätzlich ebenfalls vom aktuellen Profil ab, das eine Standard-Menü-Konfiguration definiert. Aber der Benutzer kann mit Hilfe der entsprechenden Anpassen-Dialoge die Menüstruktur verändern.

Die Beschreibungen zu den einzelnen Befehlen sind hier in der Online Hilfe nach den Kommando-Kategorien sortiert, wie sie auch in den Anpassen-Dialogen verwendet werden.

12.2 Lesezeichen

Die Befehlskategorie "Lesezeichen" enthält die Befehle für die Handhabung von Lesezeichen in Texteditoren.

12.2.1 Befehle der Kategorie Lesezeichen

Das Plug-In 'Bookmarks' stellt die Lesezeichen-Funktionalität für Texteditoren bereit. Lesezeichen können Zeilen hinzugefügt werden, um das Navigieren in langen Programmen zu erleichtern.

Die im folgenden beschriebenen Befehle ermöglichen das Setzen von und Navigieren mit den Lesezeichen. Sie sind standardmäßig im Untermenü 'Lesezeichen' im Bearbeiten-Menü enthalten. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

12.2.2 Lesezeichen ein-/ausschalten

Symbol:

Mit diesem Befehl (Kategorie Lesezeichen) kann in einem Texteditor ein Lesezeichen in der aktuellen Zeile gesetzt werden bzw. ein gesetztes Lesezeichen wieder entfernt werden. Ein gesetztes Lesezeichen wird als cyan-blaues Quadrat vor der Zeile angezeigt.

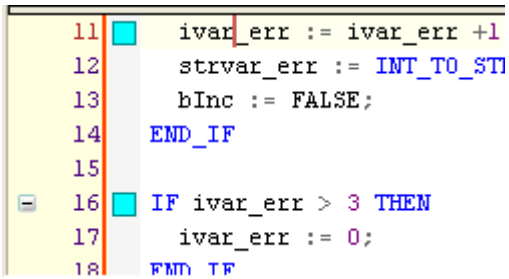


Abb. 72 Beispiel, Lesezeichen im ST-Editor

12.2.3 Nächstes Lesezeichen

Symbol:

Mit diesem Befehl (Kategorie Lesezeichen) kann im Texteditor zum nächsten Lesezeichen (nach unten) gesprungen werden.

12.2.4 Vorheriges Lesezeichen

Symbol:

Mit diesem Befehl (Kategorie Lesezeichen) kann im Texteditor zum vorherigen Lesezeichen (nach oben) gesprungen werden.

12.2.5 Lesezeichen löschen

Symbol:

Mit diesem Befehl (Kategorie Lesezeichen) werden alle Lesezeichen im aktiven Editorfenster gelöscht.

12.3 Haltepunkte

Die Befehlskategorie "Haltepunkte" enthält die Befehle für die Handhabung von Haltepunkten (Breakpoints) zu Debugging-Zwecken im Online-Betrieb.

12.3.1 Befehle der Kategorie Haltepunkte

Das 'Breakpoints' Plug-In stellt Hilfsfunktionen für das Debugging (Fehlersuche) eines Applikationsprogramms bereit. Zum Einen können **Haltepunkte** (**Breakpoints**) definiert und verwendet werden, an denen das Applikationsprogramm bei der Ausführung gestoppt wird. Zum anderen besteht auch die Möglichkeit, das Programm in definierten Schritten auszuführen (**Stepping**).

Die entsprechenden Befehle stehen in Befehlskategorie 'Haltepunkte' bereit. Standardmäßig sind sie im Online-Menü enthalten. Falls erforderlich, können Sie die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfigurieren.

Folgende Befehle dienen dem Anlegen von Haltepunkten und der schrittweisen Ausführung eines Programms:

- Haltepunkt umschalten
- Neuer Haltepunkt
- Haltepunkt-Positionen
- Einzelschritt
- Prozedurschritt
- Ausführen bis Rücksprung
- Nächste Anweisung festlegen
- Ausführen bis Cursor
- Nächste Anweisung anzeigen

Beachten Sie auch den Befehl Haltepunkte (Kategorie Ansicht), der standardmäßig im Ansicht-Menü liegt und den Haltepunkte-Dialog öffnet.

12.3.2 Haltepunkt umschalten

Default Shortcut: <F9>

Dieser Befehl (Kategorie Haltepunkte) dient grundsätzlich dazu, zwischen dem aktiven und inaktiven Zustand eines Haltepunkts hin- und herzuschalten. Allerdings bewirkt er auch das Setzen eines neuen Haltepunkts, wenn an der aktuellen Haltepunkt-Position noch keiner gesetzt ist. (Beachten Sie zum Definieren eines neuen Haltepunkts auch die Möglichkeit, dies im Dialog Neuer Haltepunkt zu tun.)

Wenn bereits ein aktiver Haltepunkt vorliegt, wird dieser inaktiviert. Wenn bereits ein inaktiver Haltepunkt vorliegt, wird dieser wieder aktiviert.

Hinweis: Jeder aktive Haltepunkt wird zum inaktiven, wenn der Online-Modus verlassen wird und danach wieder eingeloggt wird.

Ein aktiver Haltepunkt ist mit dem ●-Symbol gekennzeichnet, ein inaktiver mit dem ○Symbol.

Hinweis: Beachten Sie auch die entsprechenden Funktionen im Haltepunkte-Dialog.

12.3.3 Neuer Haltepunkt

Symbol:

Dieser Befehl (Kategorie Haltepunkte) dient dazu, einen neuen Haltepunkt im Projekt zu definieren. Es **ist nicht relevant, wo der Cursor gerade positioniert ist**, es öffnet sich der Dialog 'Neuer Haltepunkt', wo im Unterdialog **Position** eine der möglichen Haltepunkt-Positionen im gesamten Projekt ausgewählt werden kann, und im Unterdialog **Bedingung** bestimmte Bedingungen für das Wirksamwerden des Haltepunkts definiert werden können.

Hinweis: Um einen Haltepunkt an der aktuellen Cursor-Position zu setzen kann auch der Befehl Haltepunkt umschalten bzw. die entsprechende Funktion im 'Haltepunkte'-Dialog verwendet werden.

Location

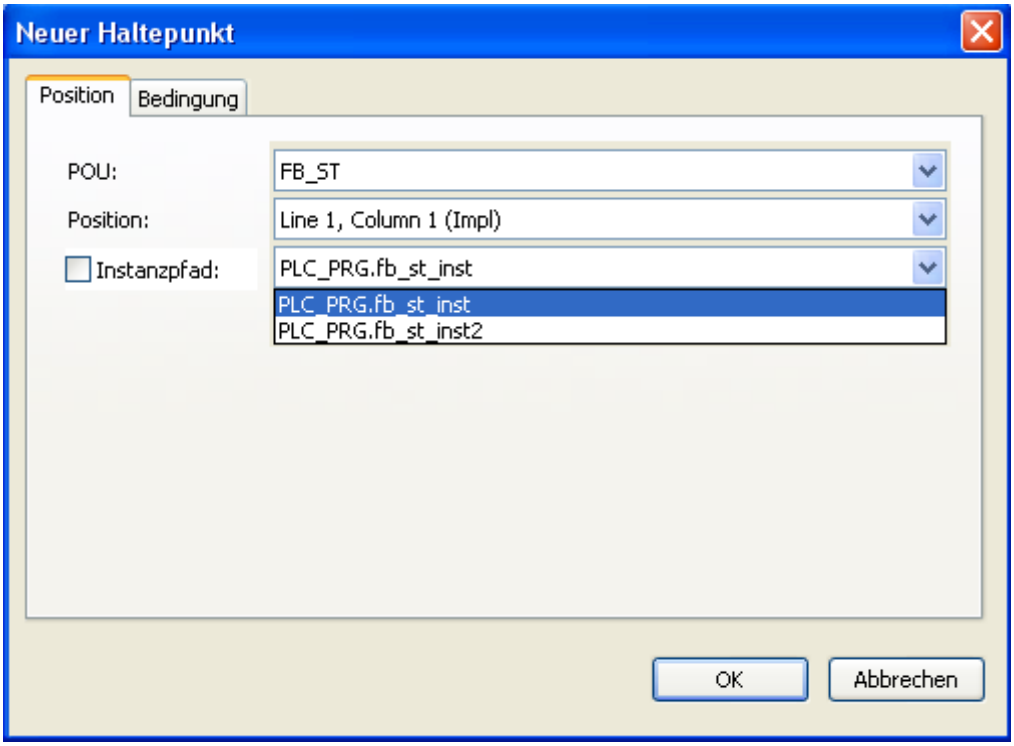


Abb. 73 Neuer Haltepunkt'-Dialog, Position

POU: Die Auswahlliste bietet alle POU-Objekte des Projekts an. Wählen Sie das aus, in dem der Haltepunkt gesetzt werden soll.

Position: Die Auswahlliste bietet alle möglichen Haltepunkt-Positionen der oben eingestellten POU an. Abhängig vom Editortyp sind diese Positionen in der Form von Zeilen- und Spaltennummern (Texteditor) oder als Netzwerk- oder Elementnummern. Im Falle von Funktionsblöcken, wird zusätzlich ein "(Impl)" angezeigt und Sie müssen entscheiden, ob der Haltepunkt in der **Implementation** oder in einer **Instanz** gesetzt werden soll. Wenn er in der Implementationsversion gesetzt werden soll, lassen Sie die Option 'Instanzpfad' deaktiviert. Wenn er in einer Instanz gesetzt werden soll, aktivieren Sie die Option 'Instanzpfad', um die Instanz auszuwählen; siehe folgende Beschreibung:

Instanzpfad: Wenn die oben gewählte POU ein Funktionsblock ist, und diese Option deaktiviert ist, wird der neue Haltepunkt in der Implementationsversion des Funktionsblocks gesetzt (s.o. Position). Wenn der Haltepunkt in einer Instanz gesetzt werden soll, aktivieren Sie diese Option und wählen die gewünschte Instanz aus.

Bedingung

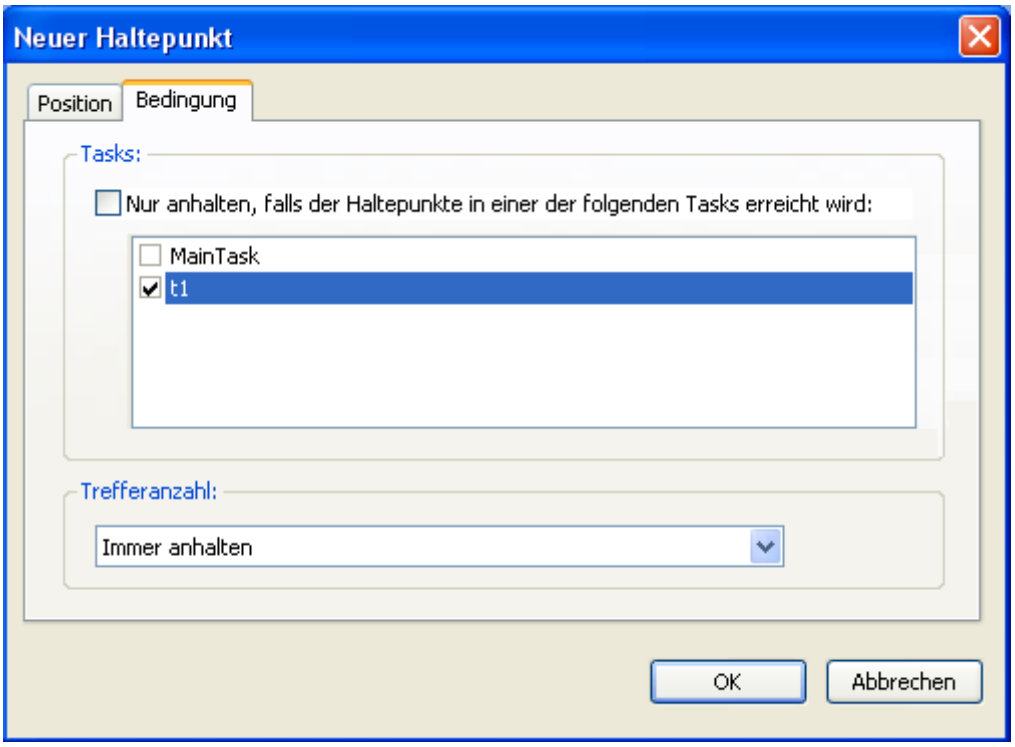


Abb. 74 'Neuer Haltepunkt'-Dialog, Bedingung

Tasks:

Nur anhalten, falls der Haltepunkt in einer der folgenden Tasks erreicht wird: Wenn diese Option aktiviert ist (Default: deaktiviert), wird der Haltepunkt nur wirksam, wenn die POU, in der er gesetzt ist, von einer der hier ausgewählten Tasks ausgeführt wird. Alle im Projekt definierten Tasks sind aufgelistet und die gewünschten können durch Setzen eines Hakens im Kontrollkästchen ausgewählt werden.

Trefferanzahl:

Immer anhalten: Das Programm stoppt immer an diesem Haltepunkt.

Alternativ kann eine Zahl angegeben werden, die sagt, wie oft der Haltepunkt getroffen worden sein muss, bevor das Programm gemäß einer der folgenden Bedingungen in der Ausführung stoppt.

- Anhalten, wenn die Trefferanzahl gleich
- Anhalten, wenn die Trefferanzahl ein Vielfaches von
- Anhalten, wenn die Trefferanzahl größer oder gleich

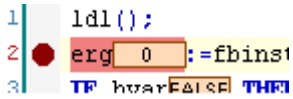
12.3.4 Haltepunkt-Positionen

Die möglichen Haltepunkt-Positionen hängen vom Typ des Editors ab. Grundsätzlich sind es jene Positionen in einer POU, an denen sich die Werte von Variablen ändern können oder an denen sich der Programmfluss verzweigt bzw. ein anderer Baustein aufgerufen wird.

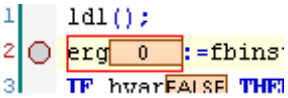
Hinweis für Haltepunkte in Methoden: Ein Haltepunkt wird immer automatisch in allen Methoden gesetzt, die aufgerufen werden können. Das heißt, wenn eine Schnittstellen-verwaltete Methode aufgerufen wird, werden auch Haltepunkte in allen Methoden der Funktionsblöcke gesetzt, die die Schnittstelle implementieren und ebenso in allen abgeleiteten Funktionsblöcken, die die Methode in Anspruch nehmen. Wenn eine Methode über einen Pointer auf einen Funktionsblock aufgerufen wrid, werden Haltepunkte in der Methode des Funktionsblocks und in allen abgeleiteten Funktionsblöcken, die die Methode beanspruchen, gesetzt.

Haltepunkt-Symbole

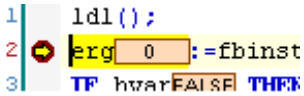
Haltepunkt im Online-Modus:



Deaktivierter Haltepunkt:



Programm-Stop an Haltepunkt:



12.3.5 Einzelschritt

Symbol:

Default Shortcut: <F11>

Dieser Befehl (Kategorie Haltepunkte) kann verwendet werden, um im Online-Modus ein Programm in Einzelschritten auszuführen (Stepping), z. B. um die Fehlersuche zu erleichtern.

Das Ausführen des Befehls bedeutet jeweils, dass ein Einzelschritt ausgeführt wird. Das Programm stoppt vor der nächsten Anweisung. Wenn erforderlich, wird zu einer anderen POU gewechselt. Wenn die aktuelle Position ein Funktions- oder Funktionsblockaufruf ist, wird vor der ersten Anweisung des aufgerufenen Bausteins gestoppt.

In allen anderen Situationen wirkt der Befehl wie 'Prozedurschritt'.

Mögliche Stop-Positionen während der Einzelschritt-Abarbeitung hängen vom Editortyp ab. Die aktuelle Position wird gelb hinterlegt angezeigt.

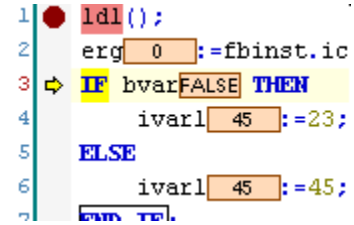


Abb. 75 Einzelschritt, Beispiel

12.3.6 Prozedurschritt

Symbol:

Default Shortcut: <F10>

Dieser Befehl (Kategorie Haltepunkte) kann verwendet werden, um im Online-Modus ein Programm in definierten Schritten auszuführen (Stepping), z.B. um die Fehlersuche zu erleichtern.

Ein einzelner Schritt wird ausgeführt. Bei einstufigen Anweisungen entspricht dies dem "Steppen" mit Befehl Einzelschritt (also von Anweisung zu Anweisung). Wenn jedoch ein Bausteinaufruf erreicht wird, bewirkt der 'Prozedurschritt', dass der aufgerufene Baustein komplett innerhalb des momentanen Schritts abgearbeitet wird. In einem Ablaufdiagramm wird eine komplette Aktion abgearbeitet.

Wenn Sie also nur einen Schritt bis zur ersten Anweisung eines aufgerufenen Bausteins ausführen wollen, müssen Sie den Einzelschritt-Befehl verwenden.

Siehe auch: Ausführen bis Rücksprung

12.3.7 Ausführen bis Rücksprung

Symbol:

Default Shortcut: <Strg>+<F11>

Dieser Befehl (Kategorie Haltepunkte) kann bei der schrittweisen Ausführung eines Applikationsprogramms (z.B. zu Debugging-Zwecken) verwendet werden.

Wenn das Programm keine Aufrufe enthält, bewirkt der Befehl 'Ausführen bis Rücksprung', dass mit dem Schritt bis zum Programmstart zurückgesprungen wird. Wenn allerdings vorher in einen aufgerufenen Baustein gesprungen wurde, bewirkt er, dass bis zur aufrufenden Anweisung zurückgesprungen wird (Hinweis: Dies ist neu im Vergleich zu »PLC Designer« V2.3). Somit führt 'Ausführen bis Rücksprung' im Falle von verschachtelten Aufrufen in der Hierarchie der Aufrufer schrittweise zurück. Dies erlaubt beispielsweise, eine Stufe zurückzuspringen um von dort in eine andere aufgerufene POU hineinzuspringen.

Siehe auch: Einzelschritt, Prozedurschritt

12.3.8 Nächste Anweisung festlegen

Symbol:

Dieser Befehl (Kategorie Haltepunkte) kann bei der schrittweisen Ausführung eines Applikationsprogramms (z. B. zu Debugging-Zwecken) verwendet werden, um die Anweisung festzulegen, die als nächstes ausgeführt werden soll. Zu diesem Zweck platzieren Sie den Cursor in dieser Anweisung und führen den Befehl aus.

12.3.9 Ausführen bis Cursor

Symbol:

Dieser Befehl (Kategorie Haltepunkte) kann (z. B. zu Debugging-Zwecken) verwendet werden, um das Programm bis zu einer temporär definierbaren Position auszuführen. Der Effekt ist der selbe wie wenn der nächste Haltepunkt an dieser Position definiert ist.

Platzieren Sie den Cursor an der gewünschten Halteposition und führen Sie den Befehl aus. Die Anweisungen die zwischen der momentanen und der festgelegten Position liegen, werden daraufhin ausgeführt.

12.3.10 Nächste Anweisung anzeigen

Symbol:

Dieser Befehl (Kategorie Haltepunkte) kann verwendet werden, um im Online Modus an die aktuelle Ausführungsposition zurück zu gelangen. Dies kann nützlich sein, wenn aus irgendeinem Grund während der schrittweisen Abarbeitung eines Programms der Cursor irgendwo anders in der Benutzeroberfläche platziert wurde. Das Fenster des entsprechenden Bausteins wird wieder in den Vordergrund geholt und der Cursor vor der nächsten auszuführenden Anweisung platziert.

12.4 Quelltextnavigation

Die Befehlskategorie 'Quelltextnavigation' wird durch verschiedene Plug-Ins mit Befehlen gefüllt. Sie enthält Befehle für die Auswertung des Aufrufbaums einer POU und zum Springen an die Stelle an der eine POU definiert ist ('BrowseProject' Plug-In), aber auch für das Ausgeben der aktuellen Querverweisliste ins Meldungsfenster ('Cross Reference View' Plug-In).

12.4.1 Gehe zur Definition

Befehle der Kategorie Quelltextnavigation - Gehe zur Definition

Das Plug-In 'BrowseProject' stellt Befehle für Befehlskategorie 'Quelltextnavigation' bereit, die dazu dienen, für einen Baustein oder eine Variable, die gerade in einem Editor bearbeitet werden, Informationen über ihre Position und Abhängigkeiten (Aufrufbaum) im Quelltext zu finden.

Standardmäßig sind diese Befehle im Bearbeiten-Menü, Untermenü "Symbol suchen" zu finden. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog verändert werden.

Gehe zur Definition

Symbol:

Dieser Befehl (Kategorie Quelltextnavigation) kann verwendet werden, wenn der Cursor gerade in einem Editor-Fenster auf einem Bezeichner steht. Er bewirkt, dass das Projekt nach der Stelle durchsucht wird, die/ die "Definition" der betreffenden Variablen bzw. des Bausteins enthält und wird den gefundenen Baustein und die Zeile/das Netzwerk im entsprechenden Editor öffnen.

Der Befehl entspricht dem "Zoom zu Baustein", das in »PLC Designer« V2.3 verwendet wird, und kann im Offline- und Online-Modus verwendet werden. Ggfs. wird der 'Online-Status wählen'-Dialog geöffnet, wo Sie wählen, ob der Baustein im Online. oder Offline-Modus geöffnet werden soll und - im Fall von Funktionsblockinstanzen - ob eine Instanz oder die Implementierung angezeigt werden soll.

Beispiele:

Der folgende Baustein enthält eine Funktionsblock-Definition ("fbinst"), einen Programmaufruf ("prog_y)" und einen Funktionsblockaufruf ("fbinst.out"):

```
VAR
fbinst:fb1;
ivar:INT;
END_VAR
prog_y();
ivar:=prog_y.y;
res1:=fbinst.out;
```

Wenn Sie den Cursor auf "prog_y" stellen, öffnet der Befehl das Programm "prog_y" in seinem Editor.

Wenn Sie den Cursor auf "fbinst" stellen, setzt der Befehl den Fokus ins Deklarationsfenster in Zeile "fbinst:fb1;"

Wenn Sie den Cursor auf "out" stellen, öffnet der Befehl den Funktionsblock "fb1" in seinem Editor.

12.4.2 Querverweisliste

Befehle der Kategorie Quelltextnavigation - Querverweise

Das Plug-In 'Cross Reference View' stellt Befehle der Kategorie Quelltextnavigation zur Handhabung einer Querverweisliste bereit.

Die Befehle sind direkt im Fenster der Querverweisliste enthalten.

Querverweise ausgeben

Symbol:

Dieser Befehl (Kategorie Quelltextnavigation) gibt die Querverweisliste, die gerade im Querverweislisten-Fenster angezeigt wird, ins Meldungsfenster aus. Das kann von Interesse sein, wenn die automatische Aktualisierung der Querverweisliste aktiviert ist.

Dieser Befehl ist auch über eine entsprechende Schaltfläche im Querverweislisten-Fenster verfügbar.

Gehe zur Definition

Symbol:

Dieser Befehl (Kategorie Quelltextnavigation) kann verwendet werden, um aus der Querverweisliste zu dem Ort im Projekt zu springen, in der die referenzierte Variable deklariert wird. Der entsprechende Deklarationseditor öffnet sich und die Variable wird dort markiert.

12.5 Übersetzen

Die Befehlskategorie "Übersetzen" enthält Befehle zum Übersetzen (Syntax-Check) der Projektdatei.

12.5.1 Befehle der Kategorie Übersetzen

Das Plug-In 'Build Commands' liefert Befehle für die Befehlskategorie 'Übersetzen', d.h. für das Kompilieren eines Applikationsproramms. Wenn nicht explizit anders lautend, beziehen sie sich jeweils auf die gerade aktive Applikation.

Die "Build"-Befehle dienen nicht nur der Code-Erzeugung sondern auch syntaktischen Prüfungen in geänderten oder allen Objekten der aktiven Applikation. Auch Offline Code-Generierung ist möglich, um Kompilierungsfehler prüfen zu können, bevor der Code auf das Gerät geladen wird.

Die Ergebnisse werden ins Meldungsfenster ausgegeben, Meldungskategorie: Übersetzen.

Die "Bereinigen"-Befehle dienen dazu, die Übersetzungsinformation, die beim letzten Download und Code-Erzeugen gespeichert wurde, auf dem Zielsystem zu löschen. Dies hat beispielsweise Einfluß auf Online Change.

Standardmäßig sind die Befehle im Menü "Erstellen" zu finden. Wenn erforderlich, kann die Menükonfiguration mit Hilfe des Anpassen-Dialogs verändert werden.

Hinweis: Die Version des zu verwendenden Compilers kann in den Projekteinstellungen festgelegt werden, Kategorie Compile Options (Übersetzungsoptionen).

12.5.2 Übersetzen

Symbol:

Default Shortcut: <F11>

Dieser Befehl (Kategorie 'Übersetzen') startet den Übersetzungsprozess für die gerade aktive Applikation. Das bedeutet, dass alle Objekte dieser Applikation syntaktisch überprüft werden. Beachten Sie, dass dabei kein Code generiert wird, wie es bei einem Einloggen auf das Zielsystem bzw. Download der Applikation der Fall ist! Der Übersetzungsvorgang wird automatisch vor jedem Einloggen mit geändertem Programm durchgeführt.

Nach Abschluss der syntaktischen Überprüfung werden eventuelle Fehlermeldungen oder Warnungen innerhalb des Meldungsfensters in Kategorie "Übersetzen" angezeigt. Wenn die Anzahl der Fehler/Warnungen 500 übersteigt, werden alle weiteren Meldungen übersprungen und eine spezielle wird über die Situation informieren. Befehle der Kategorie "Meldungsfenster" stehen zur Verfügung, um zwischen den Meldungen und zwischen Meldungen und Sourcecode zu navigieren.

Wenn das Programm seit dem letzten fehlerfreien Übersetzungsprozeß nicht mehr verändert wurde, wird es nicht neu übersetzt und die Meldung "Die Applikation ist aktuell" wird im Meldungsfenster ausgegeben. Wenn die syntaktische Prüfung dennoch wiederholt werden soll, verwenden Sie bitte den Befehl Neu übersetzen.

Hinweis: Die Version des zu verwendenden Compilers kann in den Projekteinstellungen festgelegt werden, Kategorie Compile Options (Übersetzungsoptionen).

12.5.3 Neu übersetzen

Dieser Befehl (Kategorie 'Übersetzen') startet den Übersetzungsprozeß für die gerade aktive Applikation, auch wenn sie zuletzt fehlerfrei übersetzt wurde.

Hinweis: Die Version des zu verwendenden Compilers kann in den Projekteinstellungen festgelegt werden, Kategorie Compile options.

12.5.4 Überprüfe alle Pool Objekte

Dieser Befehl (Kategorie 'Übersetzen') kann beim Erstellen eines Bibltioheksprojekts zu Testzwecken verwendet werden. Er veranlaßt einen Übersetzungslauf, also eine Syntax-Prüfung, für alle "Pool" Objekte, die projekt-global verfügbar im POUs-Fenster liegen. Beachten Sie, dass kein Übersetzungs-Code erzeugt wird, ebenso wird keine Datei mit Informationen zum Übersetzungslauf im Projektverzeichnis angelegt.

12.5.5 Code erzeugen

Dieser Befehl (Kategorie 'Übersetzen') ermöglicht eine Code-Generierung rein für Testzwecke. Es wird zwar genauso Übersetzungs-Code erzeugt wie beim Einloggen mit der Applikation auf die Steuerung, aber der Code wird nicht auf die Steuerung geladen. Somit kann der Code noch auf Übersetzungsfehler hin überprüft und korrigiert werden, bevor er im Online-Betrieb verwendet wird.

12.5.6 Erzeuge Laufzeitsystemdateien...

Dieser Befehl (Kategorie 'Übersetzen') generiert aus dem aktuellen Bibliotheksprojekt eine C-Rahmendatei und eine Schnittstellendatei im M4-Dateiformat, die als Basis für die Erstellung einer externen Bibliotheksdatei dienen.

Der Befehl öffnet den Dialog 'Laufzeitsystemdateien generieren':

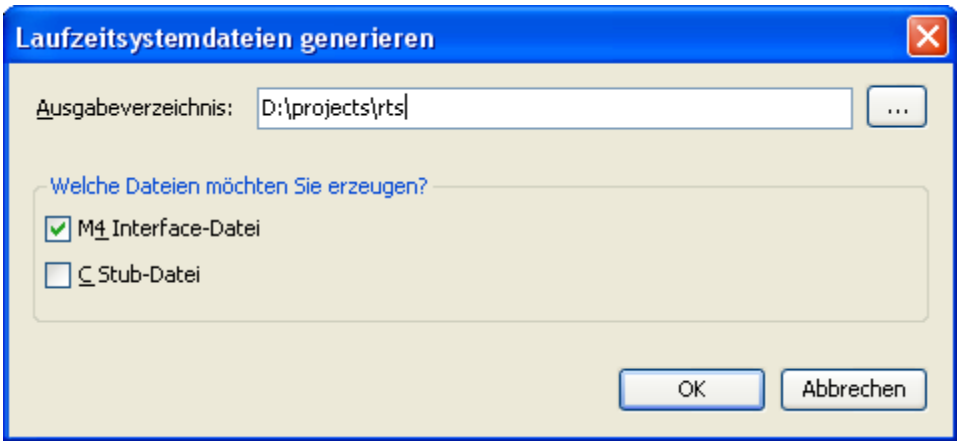


Abb. 76 Dialog 'Laufzeitsystemdateien generieren'

Ausgabeverzeichnis: Geben Sie das Verzeichnis an, in dem die Dateien angelegt werden sollen. Über Schaltfläche ... kann dazu der Standardialog zum Browsen im Dateisystem geöffnet werden.

Welche Dateien möchten Sie erzeugen?: Setzen Sie eine Haken beim gewünschten Dateityp:

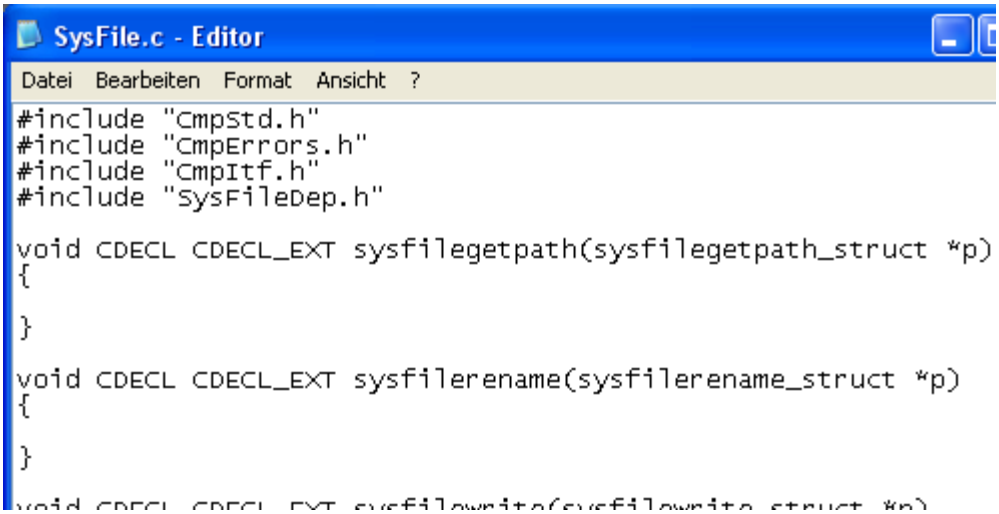
M4 Interface-Datei: Schnittstellendatei <projectname>Itf.m4 mit Definitionen. Beispiel.

SysFileItf.m4 - Editor

Datei Bearbeiten Format Ansicht ?

```
/**
 * <interfacename>SysFile</interfacename>
 * <description></description>
 *
 * <copyright></copyright>
 */

SET_INTERFACE_NAME(`SysFile')

/** EXTERN LIB SECTION BEGIN **/

#ifdef __cplusplus
extern "C" {
#endif

/**
 * <description>FileStatus</description>
 */
typedef enum
{
        FS_OK = 0,
        FS_NO_FILE = 1,
        FS_ILLEGAL_POS = 2,
        FS_FULL = 3,
        FS_EMPTY = 4,
        FS_BUFFER_FULL = 5
} FileStatus;

/**
 *      ANSI C File Access Modes
 * *    AM_READ                          Open an existing file with Rea
 * *    AM_WRITE                         Create new file with Write acc
 * *    AM_APPEND                        Open an existing file with App
 * *    AM_READ_PLUS            Open an existing file with Read/Write
 * *    AM_WRITE_PLUS           Create new file with Read/Write access
 * *    AM_APPEND_PLUS          Open an existing file with Append (rea
 *
 */
typedef enum
{
        AM_READ = 0,
        AM_WRITE = 1,
        AM_APPEND = 2.
```

C Stub-Datei: Rahmendatei für die Ausprogrammierung der Bibliothek in C. Beispiel.



12.5.7 Bereinigen

Dieser Befehl (Kategorie 'Übersetzen') löscht die Übersetzungsinformation für die gerade aktive Applikation. Diese Information wurde beim letzten Download der Applikation auf dem Zielsystem angelegt und in einer Datei (*.compileinfo) gespeichert.

Nach einem Bereinigen-Prozeß ist für die betroffene Applikation kein Online Change mehr möglich. Das Programm muss zunächst wieder komplett auf die Steuerung geladen werden.

12.5.8 Alles bereinigen

Dieser Befehl (Kategorie 'Übersetzen') löscht die Übersetzungsinformationen für alle Applikationen. Diese Information wurde beim letzten Download der Applikation auf dem Zielsystem angelegt und in einer Datei (*.compileinfo) gespeichert.

12.6 CAM

Die Befehlskategorie 'CAM' enthält Befehle für das Arbeiten im CAM-Editor.

12.6.1 CAM Befehle

Die Befehle der Kategorie 'CAM' werden vom 'CAM Editor' Plug-in bereitgestellt, welches Teil von »PLC Designer« SoftMotion V3 ist.

Standardmäßig sind Sie unter dem Menüpunkt "**CAM**" zu finden. Wenn nötig, kann über den Anpassen Dialog, Kategorie 'CAM', die aktuelle Menükonfiguration verändert werden.

Die Kommandos:

- CAM Online-Datei schreiben
- CAM-Daten in ASCII-Tabelle schreiben
- CAM Online-Datei lesen
- CAM-Daten aus ASCII-Tabelle lesen

12.6.2 CAM Online-Datei schreiben

Dieses Kommando ist standardmäßig unter dem Menüpunkt 'CAM' zu finden. Es wird benutzt um CAM Daten in einer externen Datei von proprietärem Typ ".cam" zu schreiben. Die gespeicherten CAM Daten enthalten die Beschreibung der Kurvenscheibe wie auch Position und Typ der Nocken. Sie können in ein anderes Projekt eingelesen werden indem man das Kommando CAM Online-Datei lesen ausführt.

12.6.3 CAM-Daten in ASCII-Tabelle schreiben

Dieses Kommando ist standardmäßig unter dem Menüpunkt 'CAM' zu finden. Es wird benutzt um CAM Daten in eine externe ASCII Datei (".txt") zu schreiben.

Beim Speichern der Kurvenscheibendaten werden Sie nach der Anzahl der zu speichernden Punkte, also nach der zu speichernden Datenmenge, gefragt. Diese Punkte werde dann gleichmäßig über den X Wertebereich der Kurvenscheibe verteilt. Die gespeicherten Daten enthalten die X- und Y-Positionen dieser Punkte. Information über Nocken wird nicht gespeichert.

Die gespeicherten CAM Daten können mit Hilfe des Befehls CAM-Daten aus ASCII-Tabelle lesen in ein Projekt eingelesen werden.

12.6.4 CAM Online-Datei lesen

Dieses Kommando ist standardmäßig unter dem Menüpunkt 'CAM' zu finden. Es dient dem Einlesen von Daten, die eine Kurvenscheibe beschreiben, aus einer externen Datei von proprietärem Typ ".cam", welche mit Hilfe des Kommandos CAM Online-Datei schreiben angelegt wurde.

12.6.5 CAM-Daten aus ASCII-Tabelle lesen

Dieses Kommando ist standardmäßig unter dem Menüpunkt 'CAM' zu finden. Es dient dem Einlesen von Daten, die eine Kurvenscheibe beschreiben, aus einer ASCII Datei (".txt").

Während des Einlesens der Daten werden Sie gefragt, wie viele der Kurvenpunkte beim Erzeugen der Kurvenscheibe innerhalb der Programmierumgebung berücksichtigt werden sollen.

Standardmäßig wird die Anzahl der in der Datei enthaltenen Datenpunkte vorgeschlagen, aber eine Einschränkung auf eine kleinere Menge an äquidistant verteilten Punkte kann sinnvoll sein. Da die Kurvenscheibe durch Polynom fünften Grades interpoliert wird, kann eine hohe Anzahl an Interpolationspunkten zu Oszillationen führen.

12.7 CFC

Die Befehlskategorie 'CFC' enthält Befehle für das Arbeiten im Continuous Function Chart (CFC) Editor.

12.7.1 CFC Editor Befehle

Die Befehle der Kategorie 'CFC' für das Programmieren im CFC Editor werden mit dem Plug-In 'CFC Editor' installiert.

Standardmäßig sind sie im **CFC-Menü** enthalten, das bereitsteht, wenn der CFC Editor aktiv ist. Wenn nötig, verwenden Sie den Anpassen-Dialog, Kategorie 'Menü', um die aktuelle Menükonfiguration zu verändern.

Die Befehle:

- Arbeitsblatt bearbeiten...
- Negieren
- EN/ENO
- Set/Reset:
  - Kein
  - R - Reset
  - S - Set
- Ausführungsreihenfolge:
  - An den Anfang
  - Ans Ende
  - Eins vor
  - Eins zurück
  - Nach Datenfluss anordnen
  - Topologisch Anordnen
  - Ausführungsreihenfolge setzen
- Selektierte Anschlüsse verbinden
- Anschlüsse zurücksetzen
- Nicht verbundene Anschlüsse entfernen
- Parameter bearbeiten...

12.7.2 Arbeitsblatt bearbeiten...

Das Kommando (Kategorie CFC) öffnet den Dialog **Arbeitsblatt bearbeiten**, in welchem die Größe des aktuellen CFC Arbeitsbereichs (Arbeitsblatt) geändert werden kann.

Die Größe des rechteckigen Arbeitsbereichs, der alle bestehenden CFC Elemente umfasst, wird durch Angabe von Höhe und Weite definiert, wobei der Ursprung (X: 0, Y: 0) auf der oberen linken Ecke des Editor-Fensters liegt. Höhe und Weite werden in ganzzahligen **Gittereinheiten** angegeben, wobei die Größe einer Gittereinheit vom Anwender nicht verändert werden kann. Der Wert der Höhe (Y) wächst von oben nach unten, der Wert der Breite (X) von links nach rechts.

Die maximale Größe beträgt 2048 Gittereinheiten sowohl für die Höhe als auch für die Breite.

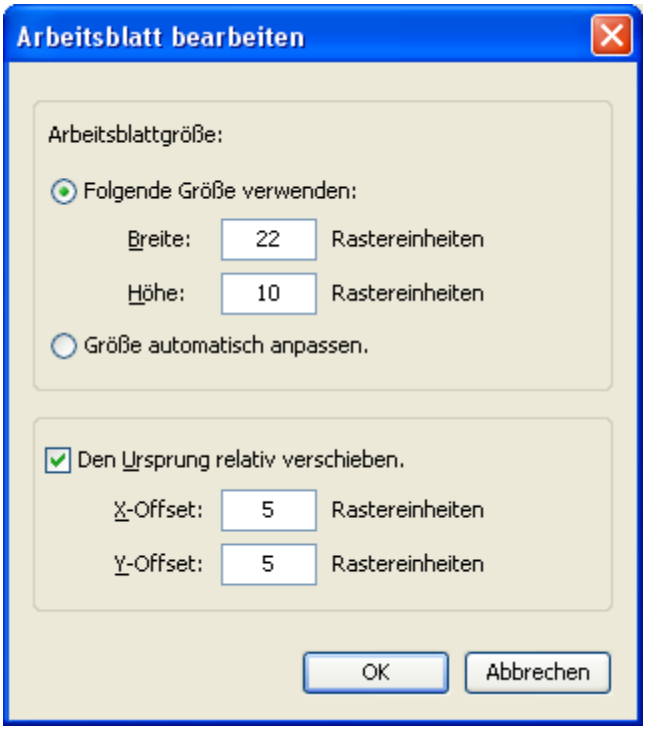


Abb. 77 Dialog Arbeitsblatt bearbeiten...

Arbeitsblattgröße:

Folgende Größen verwenden: Ist diese Option aktiviert, so wird das Arbeitsblatt gemäß der dort angegebenen Wert für Höhe und Breite dimensioniert.

Weite: enthält die aktuelle Weite in Gittereinheiten. Die Weite kann verändert werden, es ist jedoch nicht möglich einen Wert einzugeben, der kleiner ist als für die aktuell existierenden Elementen erforderlich. Der Wert der Breite (X) steigt von links nach rechts.

Höhe: enthält die aktuelle Höhe in Gittereinheiten. Die Höhe kann verändert werden, es ist jedoch nicht möglich einen Wert einzugeben, der kleiner ist als für die aktuell existierenden Elementen erforderlich. Der Wert der Höhe (Y) steigt von oben nach unten.

Größe automatisch anpassen.: Diese Option ist standardmäßig aktiviert. Die Höhe des Arbeitsblatts wird definiert durch das unterste Element, die Weite durch das am weitesten rechts positionierte Element. Der Ursprung (X=0, Y=0) ist identisch mit der oberen linken Ecke.

Den Ursprung relativ verschieben.: Nach Aktivierung dieser Option kann der Arbeitsbereich relativ zum Ursprung um die angegebenen Offset-Werte (in Gittereinheiten) vertikal oder horizontal verschoben werden. Einzige Einschränkung: Die Offsets müssen ganze Zahlen sein und dürfen die linke obere Ecke des Arbeitsbereichs nicht aus dem Fenster schieben. Bei Anschalten der Option 'Folgende Größen verwenden:' aus dem oberen Dialogteil, darf die Verschiebung die dort definierten Werte für Breite und Höhe nicht überschreiten. Anderenfalls, bei Aktivierung der Option 'Größe automatisch anpassen.', darf die Verschiebung die aktuellen Werte von Höhe und Breite überschreiten; diese werden in Folge automatisch angepasst.

X-Offset: Standardmäßig 0. Die Eingabe einer positiven Zahl bewirkt eine Verschiebung der Arbeitsfläche nach rechts und somit eventuell ihre Verbreiterung. Die Eingabe einer negativen Zahl bewirkt eine Verschiebung nach links und ist somit nur möglich, falls genug Platz zwischen dem am weitesten links platzierten Element und der linken Fenstergrenze ist.

Y-Offset: Standardmäßig 0. Die Eingabe einer positiven Zahl bewirkt eine Verschiebung der Arbeitsfläche nach unten und somit eventuell ihre Verlängerung. Die Eingabe einer negativen Zahl bewirkt eine Verschiebung nach oben und ist somit nur möglich, falls genug Platz zwischen dem obersten Element und der oberen Fenstergrenze ist.

Bei Eingabe unzulässiger Größenwerte erfolgt eine Fehlermeldung, die die oben erklärten Einschränkungen auflistet.

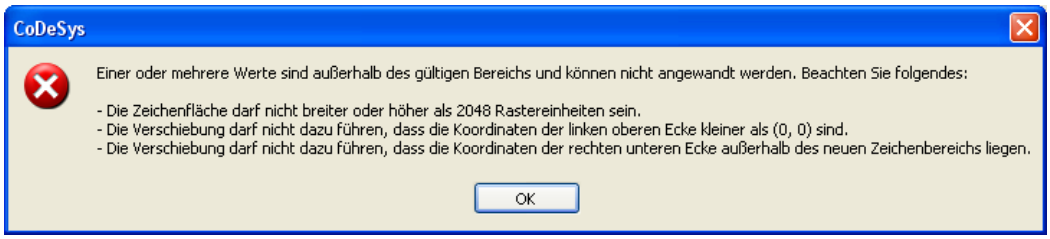


Abb. 78 Fehlermeldung bei unzulässigen Dimensionswerten

12.7.3 Negieren

Symbol:

Dieser Befehl (Kategorie CFC) wird verwendet, um Eingänge, Ausgänge, Sprünge oder RETURN-Anweisungen zu negieren. Das Symbol für die Negation ist ein kleiner Kreis auf der Verbindungslinie. Um die Negation einzufügen, wählen Sie den entsprechenden Eingangs- oder Ausgangs-Pin des jeweiligen Elements und führen den Befehl aus. Siehe CFC Editor, Cursor-Positionen für die möglichen Cursor-Positionen für die Auswahl.

Eine Negation kann durch erneutes Ausführen des Befehls wieder aufgehoben werden.

12.7.4 EN/ENO

Symbol: EN

Dieser Befehl (Kategorie CFC) wird verwendet, um dem ausgewählten Baustein (Cursor-Position 3.) einen zusätzlichen, boolschen "Enable"-Eingang EN und einen zusätzlichen, boolschen Ausgang ENO (Enable Out) zu geben.

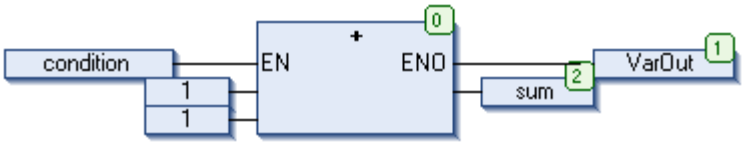


Abb. 79 ADD-Baustein mit EN/ENO

In diesem Beispiel wird ADD nur ausgeführt, wenn die boolsche Variable *condition* TRUE ist. *VarOut* wird nach der Ausführung von ADD auf TRUE gesetzt. Wenn *condition* danach wieder FALSE wird, wird ADD kein weiteres Mal ausgeführt wird und auch *VarOut* wird FALSE!

Das folgende Beispiel zeigt, wie der ENO-Wert für weitere Bausteine verwendet werden kann:

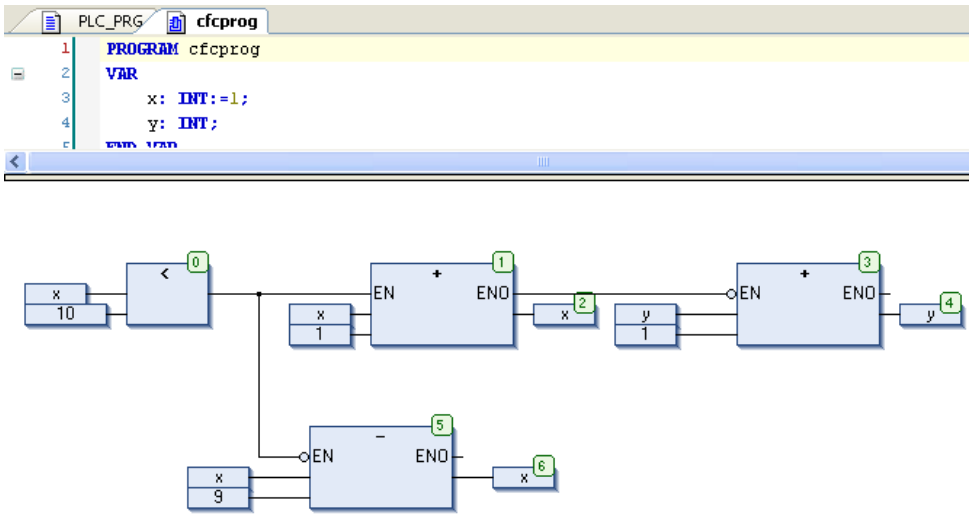


Abb. 80 Anwendung von EN/ENO, Beispiel

Initialisieren Sie für dieses Beispiel *x* mit "1". Die Nummern in der rechten oberen Ecke jedes Bausteins zeigen die Ausführungsreihenfolge im Netzwerk an.

Solange *x* kleiner als 10 (0) ist, wird es um 1 hoch gezählt (1). Sobald *x = 10*, liefert der Ausgang von LT(0) den Wert FALSE und SUB (6) und ADD (4) werden ausgeführt. *x* wird auf "1" zurückgesetzt und *y* wird um 1 hoch gezählt. LT (0) wird solange immer wieder ausgeführt, solange *x* kleiner als 10 ist. Somit zählt *y*, wie oft *x* den Wertebereich 1 bis 10 durchläuft.

12.7.5 Set/Reset

Kein

Symbol:

Dieser Befehl (Kategorie CFC) , standardmäßig im Untermenü 'Set/Reset' des CFC-Menüs, wird verwendet, um ein "Set" oder "Reset" von einem Ausgangselement zu entfernen. Selektieren Sie zu diesem Zweck den Eingangs-Pin des Ausgangselements und führen den Befehl aus. Das Set- bzw. Reset-Symbol am Eingang verschwindet daraufhin.

Siehe CFC Editor, Cursor-Positionen für möglichen Cursor-Positionen für eine Selektion.

R - Reset

Symbol:

Dieser Befehl (Kategorie CFC), standardmäßig im Untermenü 'Set/Reset' des CFC-Menüs, wird verwendet, um einem boolschen Ausgangselement ein "Reset" hinzuzufügen. Das bedeutet, dass, sobald der Eingang TRUE liefert, der Ausgang auf FALSE gesetzt wird und diesen Wert behält.

Zum Einfügen wählt man den Eingangs-Pin des Ausgangs und führt den Befehl aus.

Siehe CFC Editor, Cursor-Positionen für die möglichen Cursor-Positionen für eine Selektion.

Der zurückgesetzte Ausgang wird durch ein kleines "R"-Symbol an seinem Eingang gekennzeichnet.

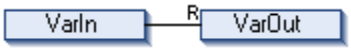


Abb. 81 Beispiel

In diesem Beispiel wird *VarOut* auf FALSE gesetzt, sobald *VarIn* TRUE liefert. *VarOut* behält diesen Wert, auch wenn *VarIn* wieder FALSE wird.

Alternativ können S (Set) oder ein None -Befehl angewendet werden, um die Set/Reset-Eigenschaften eines Ausgangs zu definieren.

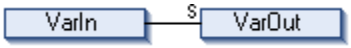
S - Set

Symbol:

Dieser Befehl (Kategorie CFC), standardmäßig im Untermenü 'Set/Reset' des CFC-Menüs, wird verwendet, um einem boolschen Ausgangselement ein "Set" hinzuzufügen. Das bedeutet, dass, sobald der Eingang TRUE liefert, der Ausgang auch auf TRUE gesetzt wird und diesen Wert behält.

Zum Einfügen wählt man den Eingangs-Pin des Ausgangs und führt den Befehl aus.

Siehe CFC Editor, Cursor-Positionen für die möglichen Cursor-Positionen für eine Selektion.



In diesem Beispiel wird *VarOut* auf TRUE gesetzt, sobald *VarIn* TRUE liefert. *VarOut* behält diesen Wert auch wenn *VarIn* wieder FALSE wird.

Alternativ können R - Reset oder ein None -Befehl angewendet werden, um die Set/Reset-Eigenschaften eines Ausgangs zu definieren.

12.7.6 Ausführungsreihenfolge

An den Anfang

Symbol:

Mit diesem Befehl (Kategorie CFC), der standardmäßig im Untermenü 'Ausführungsreihenfolge' des CFC-Menüs enthalten ist, werden im CFC Editor alle ausgewählten Elemente an den Anfang der Ausführungsreihenfolge geschoben, wobei die Reihenfolge innerhalb der ausgewählten Elemente beibehalten wird. Ebenso bleibt die Reihenfolge innerhalb der nicht ausgewählten Elemente beibehalten.

Ans Ende

Symbol:

Mit diesem Befehl (Kategorie CFC), der standardmäßig im Untermenü 'Ausführungsreihenfolge' des CFC-Menüs enthalten ist, werden alle im CFC Editor ausgewählten Elemente an das Ende der Ausführungsreihenfolge geschoben, wobei die Reihenfolge innerhalb der ausgewählten Elemente beibehalten wird. Ebenso bleibt die Reihenfolge innerhalb der nicht ausgewählten Elemente beibehalten.

Eins vor

Symbol:

Mit diesem Befehl (Kategorie CFC), der standardmäßig im Untermenü 'Ausführungsreihenfolge' des CFC-Menüs enthalten ist, werden im CFC Editor alle ausgewählten Elemente mit Ausnahme des Elements, das sich am Anfang der Ausführungsreihenfolge befindet, innerhalb der Ausführungsreihenfolge um einen Platz nach vorne verschoben.

Eins zurück

Symbol:

Mit diesem Befehl (Kategorie CFC), der standardmäßig im Untermenü 'Ausführungsreihenfolge' des CFC-Menüs enthalten ist, werden im CFC Editor alle ausgewählten Elemente mit Ausnahme des Elements, das sich am Anfang der Ausführungsreihenfolge befindet, innerhalb der Ausführungsreihenfolge um einen Platz nach hinten verschoben.

Nach Datenfluss anordnen

Mit diesem Befehl (Kategorie CFC), der standardmäßig im Untermenü 'Ausführungsreihenfolge' des CFC-Menüs enthalten ist, werden alle gerade ausgewählten Elemente im CFC Editor in Ausführungsreihenfolge (angezeigt durch die Elementnummern in der oberen rechten Ecke eines Elements) angeordnet.

Dieser Befehl wird auf alle Elemente im CFC Editor angewandt. Die Ausführungsreihenfolge wird bestimmt vom Datenfluss der Elemente und nicht von Ihrer Lage.

Der Vorteil der Reihenfolge gemäß Datenfluss ist, dass eine Ausgangsbox, die mit dem Ausgang eines Bausteins verbunden ist, in der Ausführung unmittelbar auf diesen folgt, was bei der topologischen Anordnung nicht immer der Fall ist. Die topologische Reihenfolge liefert unter Umständen also ein anderes Ergebnis als die Reihenfolge nach Datenfluss, wie man am folgenden Beispiel erkennt.

Beispiel

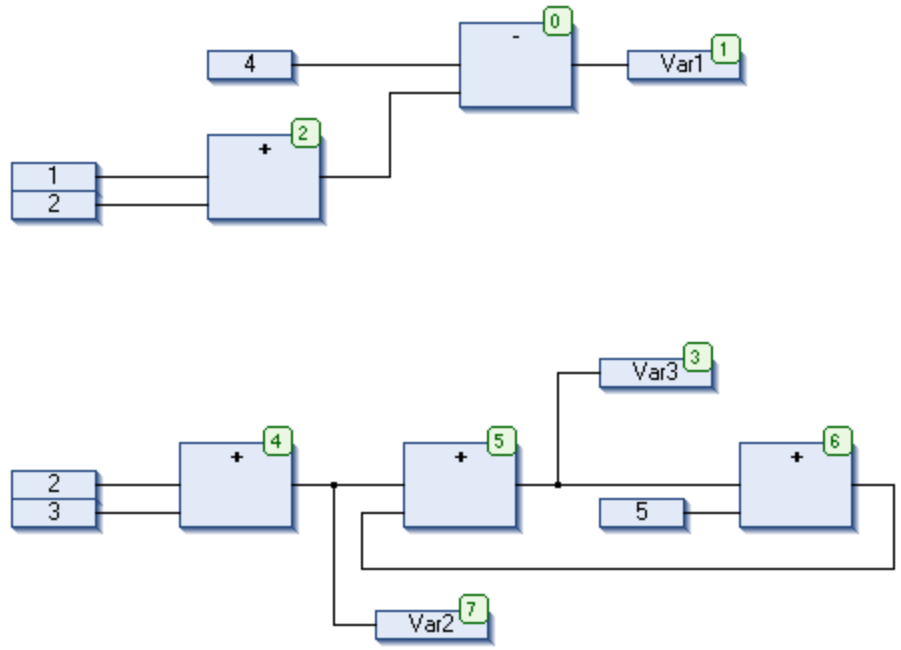


Abb. 82 Beispiel, Vorher: Topologische Ausführungsreihenfolge

Nach Anwahl des Befehls ergibt sich daraus die folgende Anordnung nach Datenfluss:

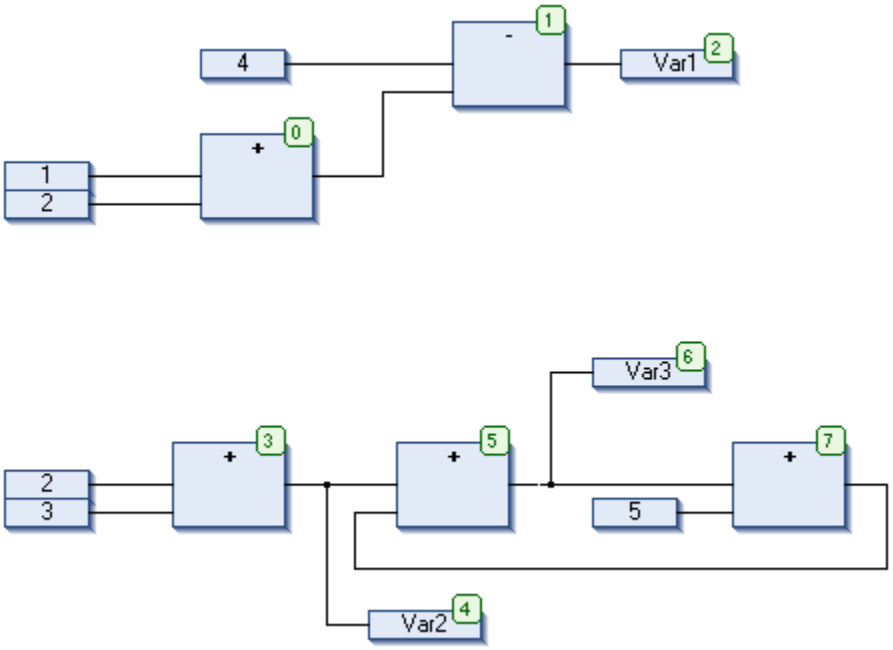


Abb. 83 Beispiel, Nachher: Ausführungsreihenfolge nach Datenfluss

Bei Anwahl des Befehls passiert zunächst intern folgendes: Alle Elemente werden topologisch sortiert. Danach wird eine neue Abarbeitungsliste zusammengestellt: Ausgehend von den bekannten Werten der Eingänge wird ermittelt, welches der noch nicht numerierten Elemente als nächstes abgearbeitet werden kann. Im oberen "Netzwerk" kann z.B. der Baustein ADD (2) sofort abgearbeitet werden, da die Werte, die an seinen Eingängen anliegen ("1" und "2") bekannt sind. Erst danach kann der Baustein SUB (1) abgearbeitet werden, da das Ergebnis von ADD bekannt sein muss, usw.. Rückkopplungen werden allerdings als letztes eingefügt. So entsteht die neue Ausführungsreihenfolge gemäß Datenfluss.

Topologisch Anordnen

Mit diesem Befehl (Kategorie CFC), der standardmäßig im Untermenü 'Ausführungsreihenfolge' des CFC-Menüs enthalten ist, werden alle gerade ausgewählten Elemente im CFC Editor in topologischer Ausführungsreihenfolge (angezeigt durch die Elementnummern in der oberen rechten Ecke eines Elements) angeordnet.

Topologische Reihenfolge bedeutet, dass die **Abarbeitung von links nach rechts und von oben nach unten** stattfindet, d.h. die Elementnummer, die die Position eines Elements in der Abarbeitungsliste anzeigt, steigt bei topologisch angeordneten Elementen von links nach rechts und von oben nach unten an. Die Lage der Verbindungen spielen keine Rolle. Es **zählt nur die Position** der Elemente.

Beim Ausführen des Befehls werden intern zunächst alle ausgewählten Elemente aus der Abarbeitungsliste herausgenommen und danach einzeln **von rechts unten nach links oben** wieder in die verbliebene Liste eingefügt. Jedes markierte Element wird dabei in der Abarbeitungsliste **vor dem topologischen Nachfolger** eingefügt und die Nummern der verbliebenen Elemente entsprechend angeglichen.

Beispiel, Topologisches Anordnen ausgewählter Elemente

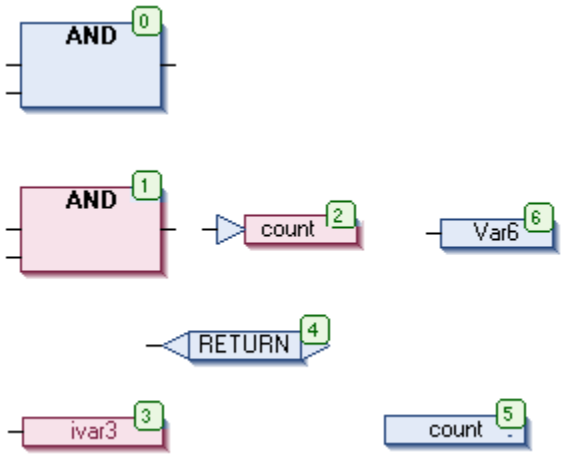


Abb. 84 Abfolge vorher

Die Elemente mit der Nummer 1, 2 und 3 sind selektiert. Wird jetzt der Befehl 'Topologisch anordnen' angewählt, werden die drei selektierten Elemente zunächst aus der Abarbeitungsliste herausgenommen. Das dann folgende Wiedereinfügen geschieht in umgekehrter Reihenfolge wie das Herausnehmen:

Zuerst wird ivar3 vor dem Label count eingeordnet, d.h. es bekommt 4, wodurch RETURN auf 3 zurückfällt.

Dann wird der Sprung count vor Var6 eingeordnet und erhält deshalb die 5. Dies bewirkt, dass Label count (hatte gerade ja die 5), Ausgang ivar3 und RETURN jeweils um eins nach unten verschoben werden.

Zum Schluss wird der AND-Baustein vor dem Sprung count wieder eingefügt und erhält deshalb 4. Dies bewirkt ein erneutes Fallen um 1 bei Label count (hatte ja gerade die 4), Ausgang ivar3 und RETURN. Es ergibt sich somit folgende neue Abarbeitungsreihenfolge:

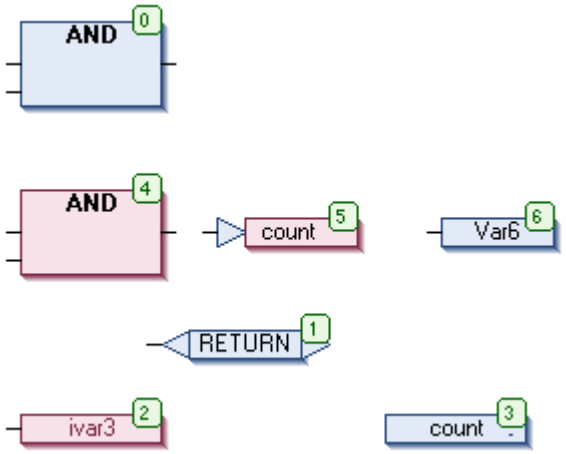


Abb. 85 Abfolge nachher

Auch ein **neues Element** wird immer vor seinem topologischen Nachfolger in die Abarbeitungsliste eingefügt.

Ausführungsreihenfolge setzen

Mit diesem Befehl (Kategorie CFC), der standardmäßig im Untermenü 'Ausführungsreihenfolge' des CFC-Menüs enthalten ist, kann die Elementnummer des gerade selektierten Elements neu definiert werden, um die Position des Elements innerhalb der Ausführungsreihenfolge zu verändern.

Der Befehl öffnet den Dialog 'Parameter bearbeiten'. Die aktuelle Elementnummer wird im Feld **Momentane Ausführungsreihenfolge** dargestellt und bei **Neue Ausführungsreihenfolge** kann die gewünschte neue eingegeben werden. Die möglichen Werte sind in Klammern angegeben.

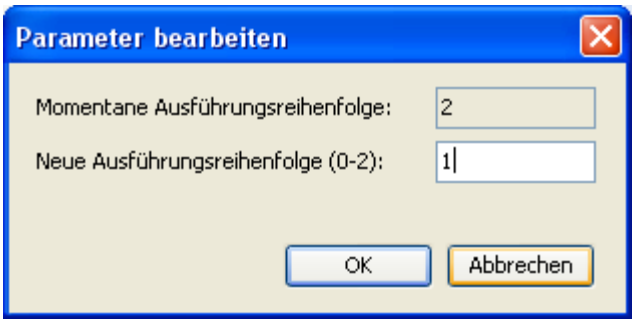


Abb. 86 Dialog 'Ausführungsreihenfolge setzen'

12.7.7 Parameter bearbeiten...

Konstante Eingangsparameter (VAR_INPUT CONSTANT) von Funktionen und Funktionsblöcken werden im CFC Editor nicht direkt dargestellt.

Beispiel:

```
FUNCTION_BLOCK fublo2
VAR_INPUT CONSTANT
fbin1:INT;
fbin2:DWORD:=24354333;
fbin3:STRING:='hallo';
END_VAR
```

Um auf diese Parameter dennoch zugreifen zu können, verwenden Sie den Befehl 'Parameter bearbeiten' (Kategorie CFC)., der den gleichnamigen Dialog öffnet. Dort werden für jeden Parameter der Name (Parameter), der Datentyp (Type) und der Initialisierungswert dargestellt.

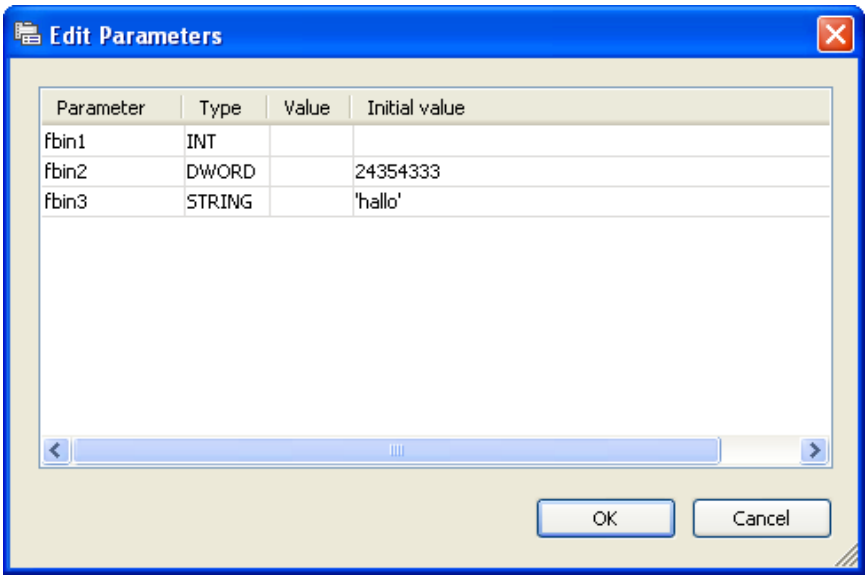


Abb. 87 Dialog 'Parameter bearbeiten'

Hier können die Werte von konstanten Eingangsparametern (VAR_INPUT CONSTANT) geändert werden. Dazu führen Sie einen Mausklick auf das jeweilige Feld in der Werte-Spalte aus, um es auszuwählen und einen weiteren Klick (oder <Eingabetaste>) um ein Eingabefeld zu öffnen. Geben Sie den gewünschten Wert ein und bestätigen mit <Eingabetaste>. **Cancel** verwirft die Eingabe. **OK** speichert alle vorgenommenen Änderungen.

Hinweis: Diese Funktionalität und die Deklaration von Variablen mit Schlüsselwort "VAR_INPUT CONSTANT" gilt nur für den CFC Editor. Im FUP Editor werden immer alle Eingangsparameter am Baustein angezeigt, egal ob sie als VAR_INPUT oder VAR_INPUT CONSTANT deklariert sind. Auch für Texteditoren macht dies keinen Unterschied.

12.7.8 Selektierte Anschlüsse verbinden

Symbol:

Das Kommando (Kategorie CFC) ist nur dann ausführbar, wenn genau ein Ausgang und mehrere Eingänge selektiert sind. Beim Aufruf wird eine Verbindung vom Anschluss des Ausgangs zu den Anschlüssen der Eingänge hergestellt.

12.7.9 Anschlüsse zurücksetzen

Symbol:

Wenn von einem Baustein Eingangs- oder Ausgangs-Pins entfernt wurden, beispielsweise weil sie nicht benutzt wurden, oder wenn sich die Schnittstelle eines Bausteins geändert hat, kann dieser Befehl (Kategorie CFC) verwendet werden, um alle Ein- und Ausgänge des Bausteins, die in seiner Implementation definiert sind, wieder darzustellen. Außerdem kann das Kommando dazu benutzt werden, die Parameter vom Typ VAR_IN_OUT eines Funktionsblocks anzuzeigen, die standardmäßig versteckt sind.

Im folgenden Beispiel wurde der Eingangs-Pin fbin2 der Funktionsblock-Instanz im CFC Editor gelöscht, weil er nicht verwendet wird. Wenn das Instanz-Element angewählt wird und der Befehl "Anschlüsse zurücksetzen" ausgeführt wird, wird fbin2 wieder dargestellt.

```
1  FUNCTION_BLOCK fb1
2  VAR_INPUT
3      fbin1:INT;
4      fbin2:INT;
5  END_VAR
6  VAR_OUTPUT
7      fbout1:INT;
8  END_VAR
9  VAR
10     fbvar:INT;
```

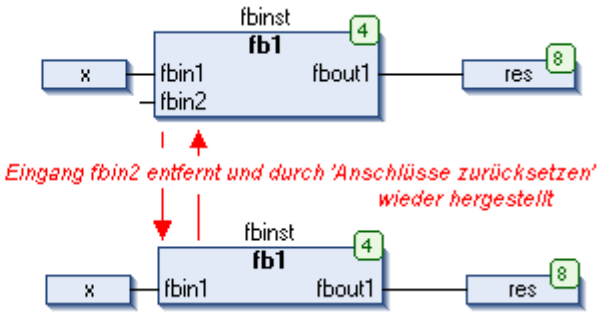


Abb. 88 Anschlüsse zurücksetzen, Beispiel

12.7.10 Nicht verbundene Anschlüsse entfernen

Symbol:

Das Kommando (Kategorie CFC) entfernt im aktuell ausgewählten Editor die nicht verbundenen Anschlüsse aus dem Aufruf von Programm, Funktionsblock oder nicht lokalen Aktionen. Aus Aufrufen von Funktionen, Methoden oder Operatoren werden die unverbundenen Anschlüsse jedoch nicht entfernt, da dies zu einer ungültigen Syntax führen würde.

12.8 CNC

Die Befehlskategorie 'CNC' enthält Befehle für das Arbeiten im CNC Editor.

12.8.1 Befehle der Kategorie CNC

Die Befehle der Kategorie **CNC** werden vom Plug-In CNC-Editor zur Verfügung gestellt, das Teil von »PLC Designer« SoftMotion V3 ist.

Standardmäßig sind die Befehle unter dem Menü **CNC** zu finden. Man kann auch den Dialog **Anpassen...** öffnen. In der Kategorie **CNC** kann dort die aktuelle Menükonfiguration angesehen oder geändert werden.

Die Befehle:

- Vorverarbeitete Bahn anzeigen
- Interpolationspunkte anzeigen
- Satzunterdrückung
- Raster anzeigen
- Endpunkte anzeigen
- Bahnansicht scrollen
- Programm verschieben
- Programm skalieren
- Programm drehen
- Richtung umkehren
- CNC-Programm neu nummerieren
- Programm in ASCII-Datei schreiben
- Programm aus ASCII-Datei laden
- Von DXF-Datei importieren
- Dynamik analysieren
- CNC-Info
- Variablen setzen

12.8.2 Vorverarbeitete Bahn anzeigen

Button:

Ist diese CNC-Option aktiviert, wird die vorverarbeitete Bahn im CNC-Editor angezeigt. Die ursprüngliche Bahn wird im Hintergrund hellgrau dargestellt und kann geändert werden. Die Berechnung der Vorverarbeitung wird bestimmt von den ausgewählten Funktionsblöcken der Bibliothek SM3_CNC im Dialog **CNC-Einstellungen** im Tabellenreiter **Bahnvorverarbeitung** ab. Dort kann ein Satz von Funktionsblöcken für die Berechnung der vorverarbeitenden Bahn wie SMC_ToolCorr, SMC_RoundPath, SMC_SmoothPath oder SMC_AvoidLoop ausgewählt werden. Auf diese Weise kann die Auswirkung von in Reihe geschalteten bahnvorverarbeitenden Bausteinen simuliert werden.

Hinweis: Diese Option hat keine Auswirkung auf die Code-Generierung.

Im Fall der Code-Generierung für die SMC_OutQueue wird die Bahnvorverarbeitung immer berechnet, unabhängig vom Status der Option.

12.8.3 Interpolationspunkte anzeigen

Button:

Ist diese CNC-Option aktiviert, wird die vorverarbeitete Bahn mit Interpolationspunkten angezeigt. Die Zykluszeit der Interpolation muss im Dialog CNC-Einstellungen im Tabellenreiter **Vorinterpolation** festgelegt werden. Sie sollte mit der Zykluszeit der aufrufenden Task übereinstimmen. Standardmäßig ist der Wert auf 20000 µs gesetzt.

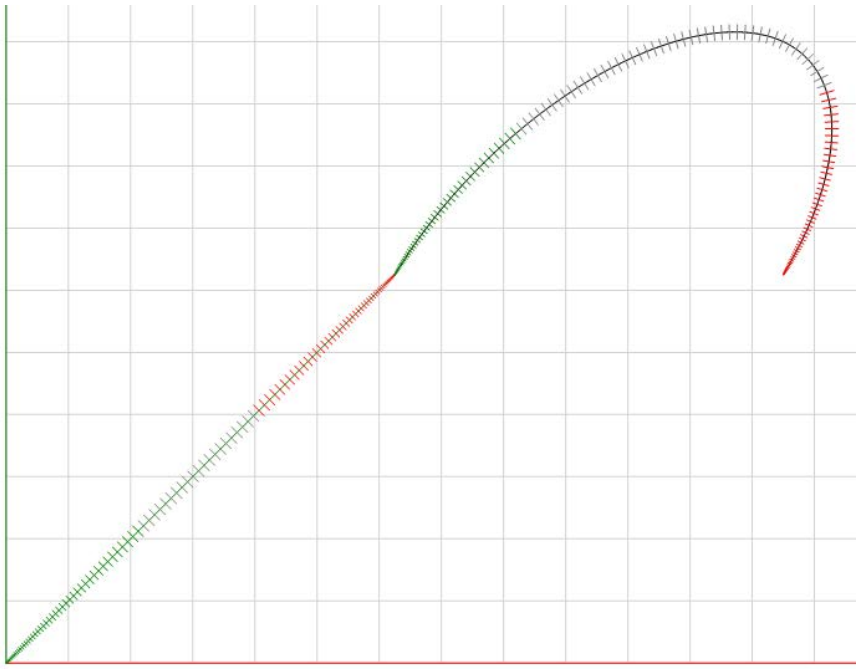


Abb. 89 Bahn mit Interpolationspunkten

Jeder Interpolationspunkt wird mit einer Linie quer zur Bewegungsrichtung dargestellt. Die Länge der Linie ist proportional zur aktuellen Bahngeschwindigkeit. Auf diese Weise ist eine grobe Abschätzung des Verhaltens der Geschwindigkeit möglich. (Schnell = langer Abstand und lange Linie, langsam = kleiner Abstand und kurze Linie). Ist der Interpolator in der Beschleunigung, dann ist die Linie grün, in Verzögerung rot und ansonsten grau.

Abhängig von der Option **Vorverarbeitete Bahn anzeigen** basieren die Interpolationspunkte auf der vorverarbeiteten Bahn, wenn die Option aktiviert ist, oder auf der ursprünglichen Bahn, wenn nicht.

12.8.4 Satzunterdrückung

Button:

Ist diese CNC-Option aktiviert, werden alle Linien des Texteditors, die mit "/" starten ignoriert. Dieser Befehl ist in Tabellenansicht nicht verfügbar.

12.8.5 Raster anzeigen

Button:

Ist diese Option im CNC-Menü aktiviert, dann ist im CNC-Editor zusätzlich ein Raster sichtbar. Die Abstände der Rasterlinien hängen vom Zoomfaktor ab. Die Größe des Rasters hängt von der Ausdehnung der Bahn ab.

12.8.6 Endpunkte anzeigen

Ist diese CNC-Option aktiviert, werden alle Bahnelemente mit Endpunkten angezeigt.

12.8.7 Bahnansicht scrollen

Ist diese CNC-Option aktiviert, scrollt und zoomt die Bahnansicht zum selektierten Element, immer wenn eine Zeile neu im Texteditor oder im Tabelleneditor selektiert wird. Somit ist immer das zuletzt selektierte Bahnelement des Texteditors im grafischen Editor vollständig sichtbar.

12.8.8 Programm verschieben

Dieser Befehl, der Teil des Menüs CNC ist, öffnet den Dialog **Verschiebungsvektor**. Dort wird ein Vektor, mit dem das CNC-Programm verschoben wird, definiert. Werte für folgende Achsen können eingefügt werden: X, Y, Z, P, Q, U, V, W.

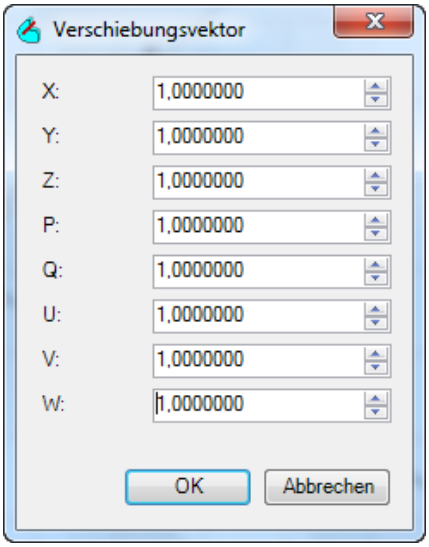


Abb. 90 Dialog Verschiebungsvektor

Bevor das erste Positionierungskommando (G1, G2, G3, G5, G6, G8, G9, G10, G11, G92) ausgeführt wird, wird ein relatives G0 Kommando mit den Koordinaten des Verschiebungsvektors eingefügt. Ist das erste Kommando ein G56 Kommando, dann beginnt das übersetzte Programm auch mit einem G56 Befehl. Die Satznummern werden automatisch zugewiesen.

Es ist nicht möglich, das CNC-Programm zu verschieben, wenn Variablen enthalten sind oder die Achsen A, B oder C verwendet werden. Wird im Programm das Koordinatensystem verschoben (G54, G55, G56), werden die Argumente angepasst und die folgenden Sätze werden nicht geändert.

12.8.9 Programm skalieren

Dieser Befehl, der Teil des Menüs CNC ist, öffnet den Dialog Programm skalieren. Hier kann der Faktor, mit dem das CNC-Programm gestreckt werden sollte und die betroffenen Achsen eingefügt werden. Nur positive Werte, die größer oder gleich 1e-3 sind erlaubt.

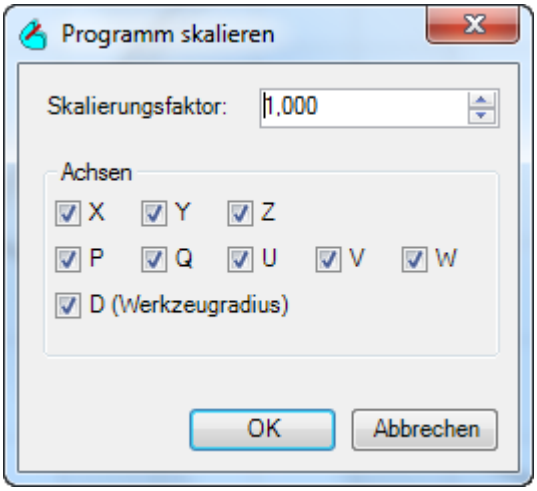


Abb. 91 Dialog Progrmm skalieren

Die Satznummern werden automatisch zugewiesen und es ist nicht möglich, ein CNC-Programm zu skalieren, das Variablen enthält oder die Achsen A, B oder C verwendet.

12.8.10 Programm drehen

Dieser Befehl, der Teil des Menüs CNC ist, öffnet den Dialog **Rotationsparameter** um das aktuelle CNC-Programm zu drehen. Dort kann der gewünschte Winkel definiert werden. Enthält das Programm nur 3D Elemente, also insbesondere keine G5-Splines, Parabeln, Ellipsen oder 2D-Kreisbögen, dann kann die Rotationsachse gesetzt werden. Standardmäßige Rotationsachse ist die z-Achse. Das Programm wird gegen den Uhrzeigersinn gedreht, wenn der Winkel positiv ist und im Uhrzeigersinn, wenn der Winkle negativ ist. Dabei wird eine Blickrichtung entgegen der Drehachse angenommen (rechte-Hand-Regel). Es ist nicht möglich ein CNC-Programm, das Variablen enthält oder die Achsen A, B oder C verwendet, zu drehen.

Die Satznummern werden automatisch zugewiesen.

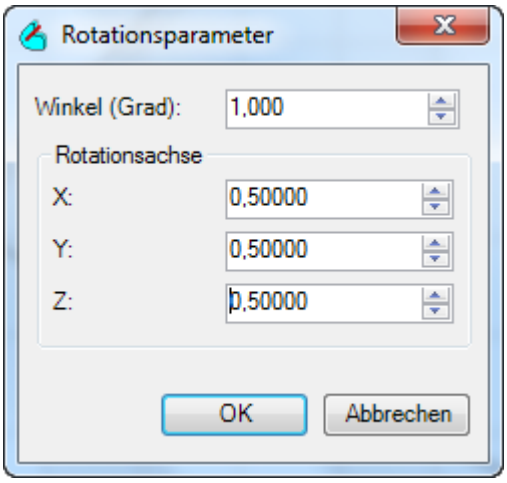


Abb. 92 Dialog Rotationsparameter

Beispiel:

Für eine Rotation um die z-Achse im Uhrzeigersinn gibt man einen negativen Winkel ein, wenn die x-Achse nach rechts und die y-Achse nach oben zeigt.

12.8.11 Richtung umkehren

Dieser Befehl, der Teil des Menüs CNC ist, kann verwendet werden, um die Programmbahn umzukehren. Enthält das CNC-Programm eines der folgenden Elemente, ist der Befehl **Richtung umkehren** nicht möglich:

- Sprünge (G20, G36 oder G37),
- Variablen,
- Satzunterdrückung,
- G56 oder G75,
- Achsen A, B oder C.

Enthält das CNC-Programm Kreisbögen, die mit Öffnungswinkel (Parameter K bei 2D-Elementen oder Parameter T bei 3D) definiert sind, kann der Befehl keinen exakten, umgekehrten Bogen generieren. G41/G42-Blöcke werden im Umkehr-Programm ausgetauscht. Somit wird aus G41 G42 und aus G42 G41 und die Richtung der Korrektur wird ausgetauscht.

Die Satznummern werden automatisch zugewiesen.

12.8.12 CNC-Programm neu nummerieren

Button:

Dieser Befehl, der Teil des Menüs CNC ist, wird verwendet um die Satznummern des Bahnprogramms im Texteditor (N<Nummer>) automatisch zurecht zu trimmen. Jeder Satz ohne Satznummer bekommt eine Nummer. Die Nummern beginnen mit 0 und werden jeweils um 10 inkrementiert. Sprungziele (L-Code in G20 Sätzen) werden ebenfalls automatisch korrigiert. Dieses Kommando ist nicht im der Tabellenansicht verfügbar.

12.8.13 Programm in ASCII-Datei schreiben

Dieser Befehl, der Teil des Menüs CNC ist, wird verwendet um ein aktuelles Programm in eine Datei im ASCII-Format zu schreiben. Mögliche Endungen sind ".txt", ".gcode" oder".cnc". Existiert die festgelegte Datei bereits, wird nach einer Bestätigung gefragt.

12.8.14 Programm von ASCII-Datei laden

Dieser Befehl, der Teil des Menüs CNC ist, wird verwendet um ein CNC-Programm zu laden, das vorher im ASCII-Format gespeichert wurde. Der Standard-Dialog um eine Datei zu öffnen, stellt eine Liste aus Dateien zur Verfügung, um die gewünschte auszuwählen. Mögliche Endungen sind ".txt" oder ".cnc", standardmäßig wird ".gcode" erwartet. Mit Klick auf die Schaltfläche **Öffnen** und bestätigen wird die ASCII-Datei geladen und in den CNC-Editor geladen, wobei das letzte CNC-Programm ersetzt wird.

12.8.15 Von DXF-Format importieren

Dieser Befehl, der Teil des Menüs CNC ist, wird standardmäßig verwendet, um eine DXF-Datei in den CNC-Editor unter dem aktuellen CNC-Objekt -Namen zu importieren. Der Standard-Dialog Öffnen, wo man nach der gewünschten DXF-Datei blättern kann, öffnet. Dann erscheint der Dialog **DXF-Datei importieren**.

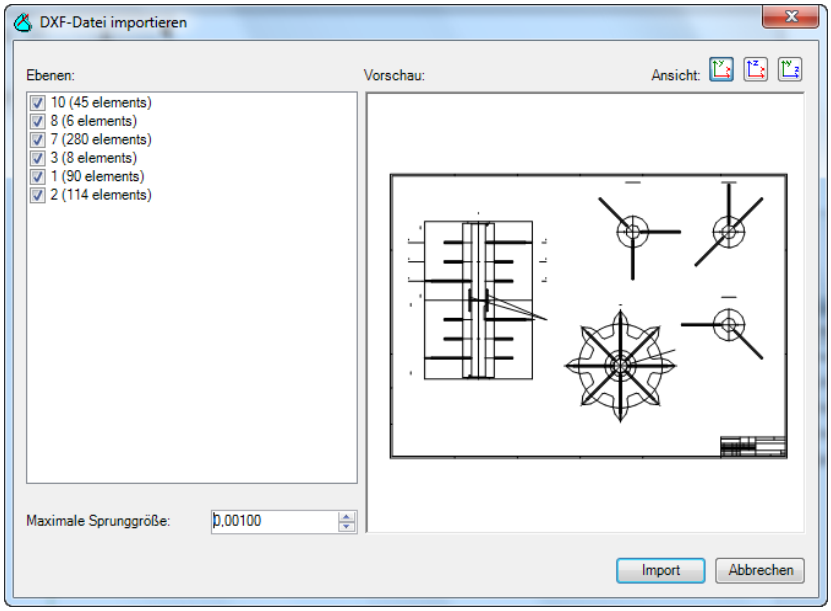


Abb. 93 Dialog DXF-Datei importieren

Im linken Teil des Dialogs wird eine Liste aller Ebenen der importierten Datei angezeigt. Jeder Ebene ist eine Checkbox zugewiesen, die es ermöglicht, Ebenen für den Import zu bestimmen. Im rechten Teil ist eine Vorschau auf die CNC-Bahn dargestellt. Mit den **Ansicht**-Schaltflächen kann zwischen x/y, x/z oder z/y Ebene gewechselt werden. **Maximale Sprunggröße** definiert den Abstand zwischen den Endpunkten der geometrischen Elemente in der DXF-Datei bis zu dem die Punkte als miteinander verbunden betrachtet werden. Mit Klicken der Schaltfläche **Import** und **Bestätigen** wird dieses Objekt in den CNC-Editor geladen und dabei das bestehende Programm ersetzt. Bitte **Rückgängig** ausführen, wenn der Import nicht wie gewünscht durchgeführt wurde.

Die Satznummern werden automatisch zugewiesen.

Der DXF-Standard wird mit folgenden Einschränkungen unterstützt:

- Splines werden nicht genau importiert.
- Keine externe Referenzen.
- Nur Ellipsen, die in der x/y-Ebene liegen.
- Nur Kreisbögen und Kreise, die eine einheitliche Skalierung in allen Achsen haben.
- Der Skalierungsfaktor einer Blockreferenz muss positv sein.
- Kein Text, Mtext oder Dimensionselemente.
- Sichtbarkeit der Elemente, Art, Farbe und Liniendicke werden ignoriert.

12.8.16 Dynamik analysieren

Dieser Befehl, der Teil des Menüs CNC ist, wird verwendet um die Dynamik der zugehörigen CNC-Bahn zu analysieren. Die Dynamik der Bahn wird in vier Diagrammen visualisiert, die eine gemeinsame Zeitachse für die selektierten Raumachsen haben: Position, Geschwindigkeit, Beschleunigung und Ruck. Mit Klick auf den Befehl öffnet der Dialog **Dynamik-Analyse**, wenn die Bahnvorverarbeitung fehlerfrei war.

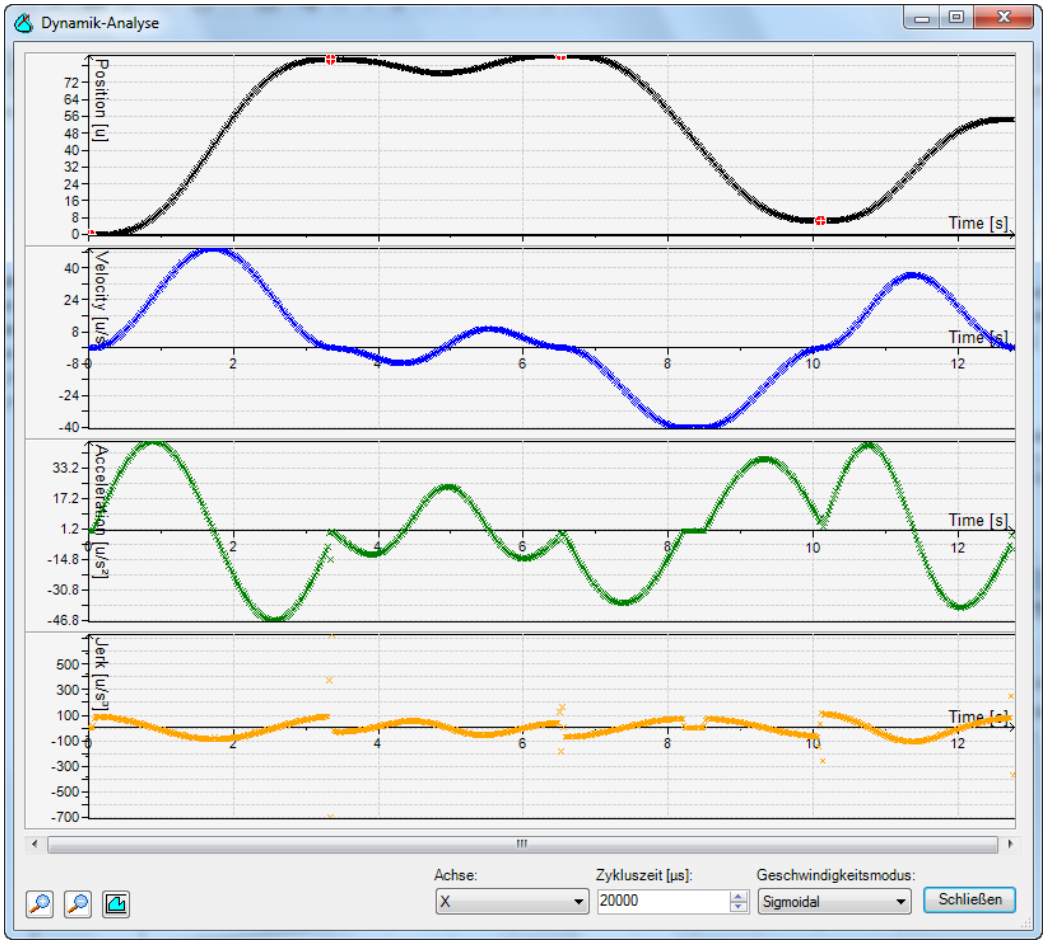


Abb. 94 Dialog Dynamik Analyse

Standardeinstellung der Achse ist x, bei Zykluszeit wird diejenige, die in **CNC-Einstellungen** definiert ist, als Initialwert verwendet, und Geschwindigkeitsmodus ist **trapezoid**. Der Zoomfaktor wird so eingestellt, dass die gesamte CNC-Bahn sichtbar ist. Geänderte Werte werden nach dem Schließen im Dialog gespeichert, so dass beim Starten des Dialogs **Dynamik Analyse** dieselben Einstellungen ausgewählt sind. Werden die CNC-Einstellungen geändert, hat dies keine Auswirkung auf die Analyse. Insbesondere der Zoomfaktor, die Scrollposition, die ausgewählte Achse und die Interpolationseinstellungen werden gespeichert.

Hinweis: Grundsätzlich werden die Einstellungen der Werte hier nur für die Dynamik-Analyse verwendet und sie haben keine weitere Auswirkung auf das CNC-Programm oder den CNC-Einstellungen.

In allen Diagrammen werden die Sätze der Datenpunkte mit kleinen Kreuzen visualisiert, so dass Sprünge sichtbar sind.

Position:

Das oberste Diagramm zeigt die Position der ausgewählten Achse über die Zeit in Abstandseinheiten. Ein roter Kreis im Diagramm markiert einen Zeitstempel, wo ein Bahnelement startet. Der Toolttip des roten Kreises informiert über die Zeilennummer im zugehörigen G-Code oder in der Tabelle.

Geschwindigkeit:

Das zweite Diagramm zeigt die Geschwindigkeit der ausgewählten Achse über die Zeit in Abstandseinheiten pro Sekunde.

Beschleunigung:

Das dritte Diagramm zeigt die Beschleunigung der ausgewählten Achse über die Zeit in Abstandseinheiten pro Sekunde im Quadrat in grün.

Ruck:

Das unterste Diagramm zeigt den Ruck der aktuellen Achse über die Zeit in Abstandseinheiten pro Sekunde in orange.

Zoom-Buttons:

Diese ersten beiden Schaltflächen werden verwendet, um den Zoomfaktor zu ändern, so dass die gemeinsamen Zeitdiagramme nach oben oder unten skaliert werden. Das gleiche wird erreicht, wenn das **Rollrad** bei gleichzeitig gedrückter **Strg**-Taste oder wenn die **+** Taste oder **-** Taste bei gedrückter **Strg**-Taste benutzt wird. Der dritte Button wird verwendet, um den Faktor so zurückzusetzen, dass die gesamte CNC-Bahn wieder sichtbar ist.

Achse:

Das Pulldown-Menü wird verwendet um eine Achse zu selektieren: X, Y, Z, A, B, C, A1(P), A2(Q), A3 (U), A4(V), A5(W), A6, Bahn. Die besondere Achse 'Bahn' zeigt die Position auf der Bahn, Bahngeschwindigkeit, Beschleunigung und Ruck.

Zykluszeit [µs]:

Ändern der Zeit mit Hilfe des Pulldown-Menüs oder durch Editieren. Dieser Wert wird als Eingang dwIpoTime des Funktionsblocks SMC_INTERPOLATOR verwendet.

Geschwindigkeitsmodus:

Zu wählen ist zwischen trapezoid oder sigmoidal. Dieser Wert wird als Eingang iVelMode des Funktionsblocks SMC_INTERPOLATOR verwendet.

12.8.17 Variablen setzen

Dieser Befehl, der Teil des Menüs CNC ist, wird verwendet um die Werte der CNC-Variablen im Offline-Modus zu setzen. Bei der Verwendung dieses Befehls öffnet der Dialog **Offline-Werte der Variablen**. Alle Variablen der zugehörigen CNC-Bahn sind dort in einer Tabelle aufgelistet. In der linken Spalte sind alle Variablen und in der rechten die aktuellen Werte aufgelistet. Die Werte können mit einem Klick auf das Tabellenfeld editiert werden. Es öffnet dann ein Inline-Editor. Mit Klick auf den **OK**-Button werden gültige Werte akzeptiert. Variablen, die in grauer, kursiver Schriftart dargestellt werden, werden im CNC-Programm nicht mehr länger verwendet.

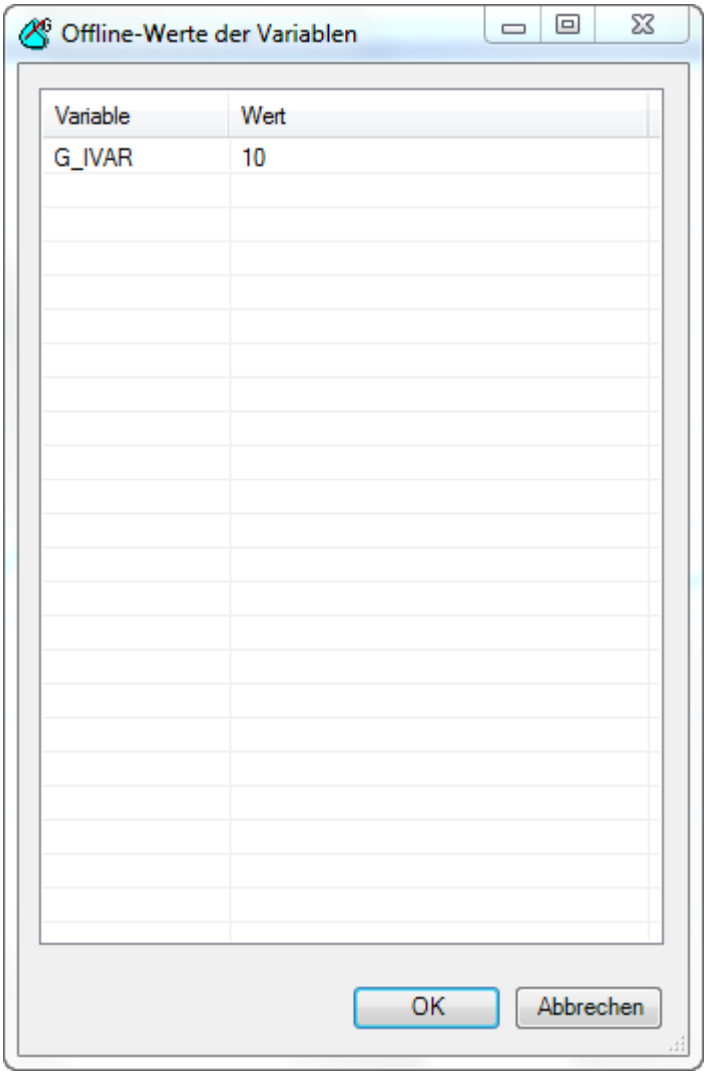


Abb. 95 Offline-Werte der Variablen

12.9 Zwischenablage

Die Befehlskategorie 'Zwischenablage' enthält Befehle für das Handhaben von Inhalten zwischen Projekteditoren und Zwischenablage.

12.9.1 Befehle der Kategorie Zwischenablage - Bearbeiten Befehle

Das Plug-In 'Edit Commands' stellt Befehle für Befehlskategorie 'Zwischenablage' bereit, die für das Anordnen und Kopieren von Objekteinträgen oder Zeichenfolgen in Objektbäumen bzw. Editoren verwendet werden können.

Standardmäßig sind diese Befehle im Bearbeiten-Menü enthalten. Falls erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

Die Befehle:

- Ausschneiden
- Kopieren
- Einfügen
- Löschen
- Alles auswählen

12.9.2 Ausschneiden

Symbol:

Default Shortcut: <Strg>+<X> oder <Umsch>+<Entf>

Dieser Befehl (Kategorie 'Zwischenablage') überträgt die aktuelle Auswahl (Objekt, Zeichenfolge) in die Zwischenablage und **entfernt sie von der bisherigen Position** im Objektbaum oder Editor. In Objektbäumen, wie z.B. im POUs- oder Geräte-Fenster bezieht sich dies auf das gerade ausgewählte Objekt. Mehrfachauswahl ist möglich.

Beachten Sie, dass nicht alle Editoren den Befehl unterstützen und dass die Verwendung in manchen Editoren eingeschränkt sein kann.

Die Art der Auswahl hängt vom Editor ab: Beispielsweise kann es in Texteditoren eine Zeichenfolge oder einzelnes Zeichen sein oder in grafischen Editoren ein oder mehrere Elemente, um die ein Auswahlrahmen gezogen ist.

Um den Inhalt der Zwischenablage einzufügen verwenden Sie den Befehl Einfügen.

Um eine Auswahl in die Zwischenablage zu übertragen, ohne sie von der aktuellen Position zu entfernen, verwenden Sie den Befehl Kopieren.

Um eine Auswahl zu entfernen, ohne die Zwischenablage zu verändern, verwenden Sie den Befehl Löschen.

12.9.3 Kopieren

Symbol:

Default Shortcut: <Strg>+<C>

Für diesen Befehl gilt die Beschreibung des Befehls Ausschneiden mit der Ausnahme, dass beim Kopieren die Auswahl **nicht von der aktuellen Position entfernt** wird.

12.9.4 Einfügen

Symbol:

Default Shortcut: <Strg>+<V>

Dieser Befehl (Kategorie 'Zwischenablage') fügt den Inhalt der Zwischenablage an der aktuellen Cursor-Position ein.

'Einfügen' wird nicht von allen Editoren unterstützt bzw. kann in manchen nur eingeschränkt verwendet werden. In grafischen Editoren wird der Befehl nur unterstützt, wenn durch das Einfügen eine korrektes Konstrukt entsteht.

In Objektbäumen, wie z.B. im POUs- oder Gerätefenster bezieht sich der Befehl auf das gerade ausgewählte Objekt. Mehrfachauswahl ist möglich. Abhängig von der aktuellen Position, z.B. im Gerätebaum, wird ev. ein Dialog geöffnet, in dem man entscheiden kann, ob das Objekt von der Zwischenablage oberhalb oder unterhalb eingefügt werden soll.

Um eine Auswahl in die Zwischenablage zu übertragen, ohne sie von der aktuellen Position zu entfernen, verwenden Sie den Befehl Kopieren.

Um eine Auswahl zu entfernen, ohne die Zwischenablage zu verändern, verwenden Sie den Befehl Löschen.

12.9.5 Löschen

Symbol:

Default Shortcut: <Entf>

Dieser Befehl (Kategorie 'Zwischenablage') löscht die Auswahl aus dem Editor bzw. Objektbaum. Er verändert nichts am Inhalt der Zwischenablage.

In Objektbäumen, wie z.B. im POUs- oder Gerätefenster bezieht sich der Befehl auf das gerade ausgewählte Objekt. Mehrfachauswahl ist möglich.

Bezüglich des Typs der Auswahl gilt dasselbe wie für den Befehl Ausschneiden.

Um eine Auswahl aus dem Editor oder Objektbaum zu entfernen und sie gleichzeitig in die Zwischenablage einzufügen, verwenden Sie den Befehl Ausschneiden.

12.9.6 Alles auswählen

Dieser Befehl (Kategorie 'Zwischenablage') kann in Editoren verwendet werden, um den gesamten Inhalt auszuwählen.

12.10 Deklaration

Die Befehlskategorie 'Deklaration' enthält Befehle für das Arbeiten im tabellarischen Deklarationseditor.

12.10.1 Tabellarischer Deklarationseditor - Deklaration

Das Plug-In 'TabularDeclarationEditor' stellt folgende Befehle der Kategorie 'Deklaration' für das Arbeiten im tabellarischen Deklarationseditor bereit:

- Deklarationskopf editieren...
- Einfügen
- Nach unten verschieben
- Nach oben verschieben

Die Befehle sind standardmäßig im Kontextmenu und in der Werkzeugleiste des Editors zu finden. Mehr Informationen dazu finden Sie in der Beschreibung des tabellarischen Deklarationseditors.

12.10.2 Deklarationskopf editieren...

Dieser Befehl (Kategorie 'Deklaration') öffnet einen Editor für den Deklarationskopf. Mehr Informationen dazu finden Sie in der Beschreibung des tabellarischen Deklarationseditors.

12.10.3 Einfügen

Dieser Befehl (Kategorie 'Deklaration') fügt eine neue Zeile für eine Variablendeklaration hinzu. Mehr Informationen dazu finden Sie in der Beschreibung des tabellarischen Deklarationseditors.

12.10.4 Nach unten verschieben

Dieser Befehl (Kategorie 'Deklaration') verschiebt die gerade ausgewählte Deklarationszeile um eins nach unten. Mehr Informationen dazu finden Sie in der Beschreibung des tabellarischen Deklarationseditors.

12.10.5 Nach oben verschieben

Dieser Befehl (Kategorie 'Deklaration') verschiebt die gerade ausgewählte Deklarationszeile um eins nach oben. Mehr Informationen dazu finden Sie in der Beschreibung des tabellarischen Deklarationseditors.

12.11 Geräte

Die Befehlskategorie 'Devices' enthält Befehler zum Installieren/Deinstallieren (Geräte-Repository) und Handhaben von Geräte-Objekten im Geräte-Fenster.

12.11.1 Geräteobjekte

Befehle der Kategorie Geräte - Geräteobjekte

Die Befehle für das Organisieren des Gerätebaums im Projekt werden hauptsächlich in Befehlskategorie 'Geräte' bereitgestellt.

Standardmäßig sind die Befehle im Projekt-Menü enthalten. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

Die Befehle:

- Gerät anhängen...
- Gerät einfügen...
- EtherCAT XML erzeugen
- SercosIII SCI xml erzeugen
- Gerät setzen
- Gerät aktualisieren
- Gerät deaktivieren

Gerät anhängen

Dieser Befehl (Kategorie Geräte) wird verwendet, um ein Geräte-Objekt, das ein Hardware-Modul repräsentiert, **eingerückt unterhalb** des gerade selektierten Objekts im Gerätebaum einzufügen. Hinweis: Um ein Objekt auf gleicher Ebene einzufügen, muss der Befehl Gerät einfügen verwendet werden.

Der Befehl öffnet den **Dialog 'Gerät anhängen'**. Abhängig von der gerade ausgewählten Position im Gerätebaum erhält man eine Liste der an dieser Stelle einfügbaren Geräte. Als Voraussetzung müssen die Geräte auf Basis von Gerätebeschreibungsdateien **auf dem lokalen System installiert** sein, was mit Hilfe des Geräte Repository Dialogs vorgenommen werden kann.

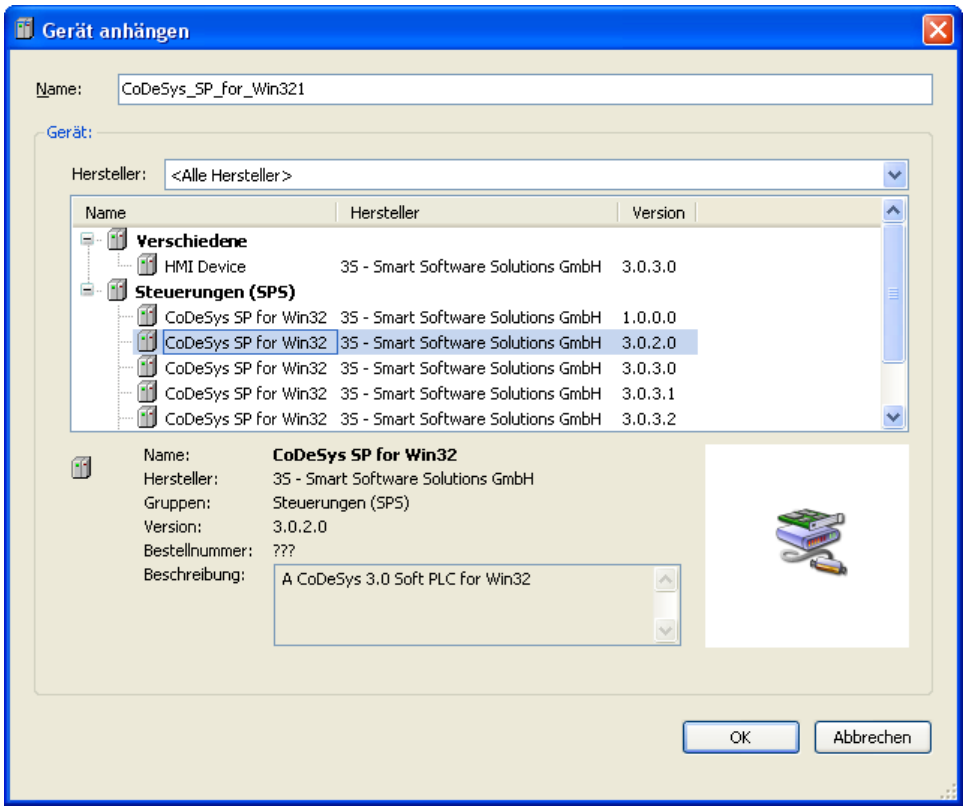


Abb. 96 Dialog 'Gerät anhängen'

Im Feld **Name** ganz oben im Dialog geben Sie ein, mit welchem Namen das Gerät im Gerätebaum erscheinen soll. Dieser Name muss ein gültiger IEC-Bezeichner sein.

Aktion:

Durch Aktivierung des entsprechenden Punktes können Sie auswählen, welche der aufgeführten Aktionen ausgeführt werden soll: Gerät anhängen, Gerät einfügen, Gerät setzen oder Gerät aktualisieren. Ihre Auswahl hat direkte Auswirkung auf die Auswahlliste im Abschnitt "Gerät" wie auch auf die Beschriftung der Schalttaste direkt links neben der Taste **Schließen** zum Schließen des Dialogs.

Gerät:

Die "scrollbare" Tabelle im Dialog 'Gerät anhängen' bietet eine Liste der Geräte, die an der aktuellen Position im Gerätebaum eingehängt werden können. Die Geräte werden mit **Name** und **Hersteller.** Bei Aktivierung des Kontrollkästchens **Alle Versionen anzeigen (nur für Experten)** werden auch alle verfügbaren Versionen der einzelnen Geräte angezeigt. Sie können gruppiert dargestellt sein, d.h. eingerückt unterhalb des Kategorienamens in der Spalte **Name**, wie beispielsweise unter Kategorie "Feldbusse" in der oben gezeigten Abbildung. Um die Gruppen zu öffnen oder zu schließen verwenden Sie die Plus- und Minuszeichen vor den Kategorienamen.

Standardmäßig werden zunächst alle verfügbaren Geräte aufgelistet (<Alle Hersteller>). Sie können jedoch Hilfe der **Hersteller**-Auswahlliste auch nur die eines bestimmten Herstellers anzeigen lassen.

Information:

Für das gerade in der Gerätetabelle des Dialogs ausgewählte Gerät wird ggfs. - wenn solche in der Beschreibungsdatei bereitgestellt ist - zusätzliche Information unterhalb der Tabelle angezeigt: Gerätename, Hersteller, Gruppen (ev. zu Kategorisierungszwecken definiert), Version, Bestellnummer und eine kurze Beschreibung. Zusätzlich kann ein gerätespezifisches Bitmap dargestellt sein.

Abhängig von der gewählten Aktion muss Ihre Auswahl durch Drücken der Taste **Gerät anhängen** / **Gerät einfügen** / **Gerät setzen** oder **Gerät aktualisieren** bestätigt werden, worauf die entsprechende Aktion ausgeführt wird. Beachten Sie, dass ohne ein erneutes Öffnen des Dialogs weitere Gerät angehängt/ eingefügt/ gesetzt oder aktualisiert werden können. Dabei kann der entsprechende Knoten im Gerätebaum jeweils durch Anklicken ausgewählt werden.

Nach Beendigen kann der Dialog durch Anklicken von **Schließen** verlassen werden.

Hinweis: Ein Gerät kann auch mit Hilfe des 'Objekt hinzufügen'-Dialogs in den Gerätebaum eingefügt werden. Er bietet allerdings nicht die Anzeige der oben beschriebenen Zusatzinformationen zum Gerät.

Gerät einfügen

Dieser Befehl (Kategorie Geräte) wird verwendet, um ein Geräte-Objekt, das ein Hardware-Modul repräsentiert, auf **gleicher Ebene** wie das gerade selektierte Objekt **vor** diesem im Gerätebaum einzufügen. Beachte: Im Unterschied dazu kann mit Befehl Gerät anhängen... ein Gerät (eingerückt) unterhalb des gerade selektierten hinzugefügt werden.

Der Befehl öffnet den **Dialog 'Gerät anhängen'** ein entsprechendes Gerät ausgewählt werden kann.

EtherCAT XML erzeugen

Mit Hilfe dieses Kommandos (Kategorie Geräte) kann die EtherCAT Konfiguration des aktuellen Projekts in XML-Format abgespeichert werden. Nach Aufruf des Kommandos werden Sie gefragt, wo die erzeugte .XML-Datei gespeichert werden soll.

SercosIII SCI xml erzeugen

Dieser Befehl (Kategorie Geräte) wird verwendet, um die SercosIII SCI Konfiguration des aktuellen Projekts im XML-Format zu speichern. Nach Aufruf des Befehls werden Sie gefragt, wo die erzeugte .XML-Datei gespeichert werden soll.

Gerät setzen

Dieser Befehl (Kategorie Geräte) wird verwendet, um ein Geräte-Objekt, das ein Hardware-Modul repräsentiert im gerade selektierten Steckplatz (Slot) einzufügen. Ein leerer Steckplatz ist durch das Symbol und den Eintrag "<Leer> (<Leer>)" gekennzeichnet . Ein belegter Steckplatz erhält das Symbol und den entsprechenden Objekteintrag.

Der Befehl öffnet den **Dialog 'Gerät anhängen'**, in dem wie unter "Gerät anhängen" beschrieben, ein entsprechendes Gerät ausgewählt werden kann. Im Falle eines belegten Steckplatzes wird mit dem Befehl das vorhandene Modul durch das neue ersetzt.

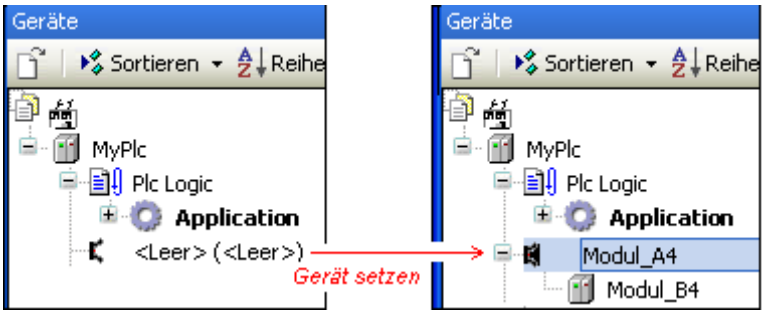


Abb. 97 Beispiel, Gerät setzen

Gerät aktualisieren

Dieser Befehl (Kategorie Geräte) wird verwendet um ein Geräteobjekt, das gerade im Gerätebaum selektiert ist, durch ein anderes zu ersetzen. So kann entweder eine andere Version eines Geräts oder ein ganz anderer Gerätetyp anstelle des bisherigen eingesetzt werden.

Wie beim normalen Einfügen eines Geräts in den Konfigurationsbaum öffnet sich der Dialog Gerät anhängen, in dem man eines der installierten und für die Position geeigneten Geräte auswählen kann. Das Gerät will anstelle des vorherigen eingefügt. Der im Gerätebaum verwendete symbolische **Gerätename** bleibt stehen, der in Klammern dahinter angegebene **Gerätetyp** ändert sich, wenn ein anderer Typ ausgewählt wurde. Wenn also nur die Geräteversion gewechselt wird, erscheint der Objekteintrag unverändert.

Der Konfigurationsbaum unterhalb des betroffenen Geräteeintrags bleibt erhalten, d.h. Submodule etc. bleiben erhalten. Auch die Konfigurationseinstellungen bleiben bestehen, außer der Gerätetyp wird gewechselt. Inkonsistenzen in der Konfiguration, die durch den Geräte-Update entstehen, werden beim nächsten Übersetzen der Applikation entdeckt und entsprechende Fehlermeldungen ausgegeben. Dies bezieht sich ggfs. auch auf implizit eingefügte Bibliotheken, die bei einem Geräte-Update nicht automatisch entsprechend entfernt werden.

Gerät deaktivieren

Das Kommando (Kategorie Geräte) wird verwendet um innerhalb der aktuellen PLC Konfiguration ein Gerät im Bus-System zu aktivieren bzw. zu deaktivieren, sofern der Bus-Treiber diese Aktion unterstützt. Ein deaktiviertes, passives Gerät wird nicht berücksichtigt und angesprochen werden. Beachten Sie, dass in manchen Bus-System das Deaktivieren eines Knoten zum Stillstand des Masters führt.

Das Kommando steht nur im Offline-Modus zur Verfügung, standardmäßig im Kontextmenü des Gerätebaums.

Der Eintrag eines deaktivierten Geräts im Baum erscheint in hellgrauer Schrift. Beim Einloggen werden deaktivierte Geräte zusätzlich mit einem roten Dreieck markiert.

Um ein Gerät erneut zu aktivieren, benützen Sie das Kommando erneut im Offline-Modus.

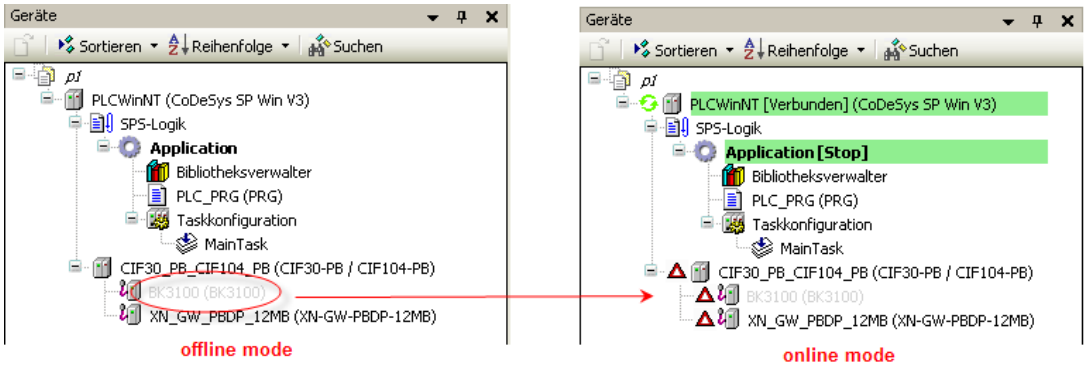


Abb. 98 Beispiel, Profibus deaktiviert

12.11.2 Geräte-Scan

Befehle der Kategorie Geräte - Geräte-Scan

Der Befehl Geräte suchen zum Scannen (Abtasten) der aktuell mit dem Projekt verbundenen Hardware-Umgebung wird in Befehlskategorie 'Geräte' bereitgestellt.

Standardmäßig sind diese Befehle im Projekt-Menü enthalten. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

Geräte suchen

Dieser Befehl (Kategorie Geräte) wird verwendet, um die aktuell zu steuernde Hardware zu scannen, d.h. ihre Struktur zu ermitteln, in einem Dialog darzustellen und dem Anwender für die Übernahme in den Gerätebaum im Projekt bereitzustellen.

Die Scan-Funktionalität ist für verschiedene Netzwerktypen unterschiedlich angelegt: Sie kann in der Steuerung fest implementiert sein oder aber durch eine Bibliothek bereitgestellt werden, die im Projekt eingebunden wird.

In jedem Fall wird für das Ausführen von Scan-Funktionen automatisch eine Verbindung zur Steuerung aufgebaut und dann wieder geschlossen. Der Gateway der SPS muss also konfiguriert sein und die Steuerung laufen. Wenn die Scan-Funktionalität fest in der SPS implementiert ist, kann das Scannen sofort durchgeführt werden. Wenn sie in einer **Bibliothek** liegt, muss **einmalig ein Login** erfolgt sein, um die Bibliothek zu laden. Wurde dies noch nicht durchgeführt, wird der Anwender beim Versuch zu scannen, dazu aufgefordert.

Der Befehl bezieht sich auf die gerade im Gerätebaum selektierte, zum Projekt verbundene Steuerung. Beispielsweise kann ein bereits eingefügter PROFINET IO Master ausgewählt werden und mit dem Befehl die ihm zugeordneten Slave-Geräte und IO-Module ermittelt werden. Der Befehl öffnet den Dialog 'Geräte scannen':

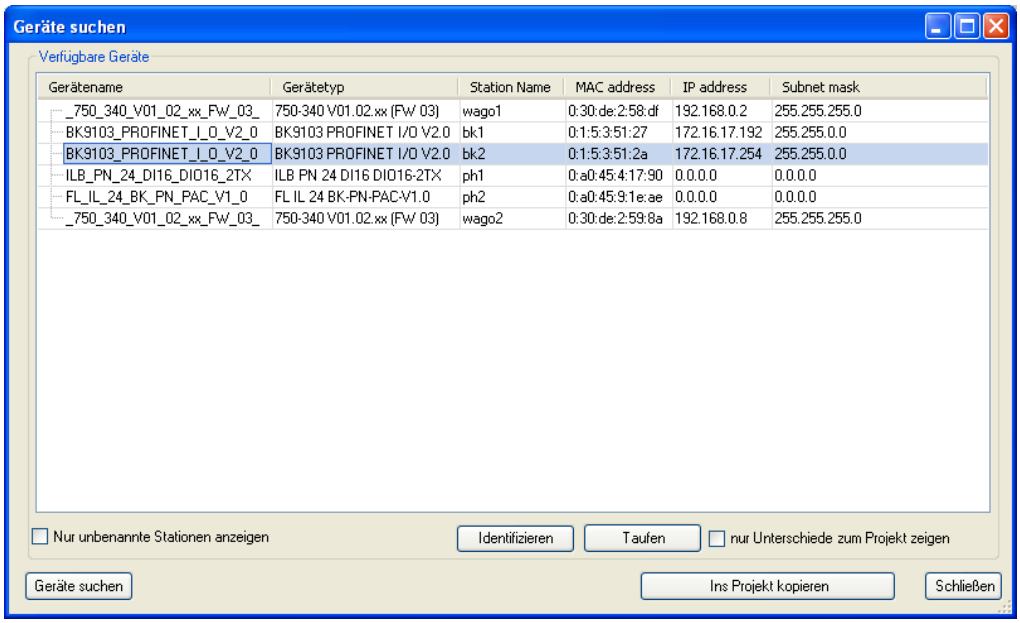


Abb. 99 Dialog 'Geräte suchen'

Der Dialog listet die bei der aktuellen Suche gefundenen Geräte und die zugehörigen Module im Fenster **Verfügbare Geräte** an.

Welche Informationen zu den Geräten angezeigt werden und welche Optionen außerdem im Dialog verfügbar sind, hängt vom jeweiligen Netzwerktyp ab.

In jedem Fall sind folgende Funktionen vorhanden:

- Im Fenster "Verfügbare Geräte" bzw. "Geräte" werden **Gerätename** und **Gerätetyp** der gefundenen Geräte angezeigt. Abhängig vom Gerätetyp können auch Stationsadresse, Ident Nr. und Status angezeigt sein.
- Über Schaltfläche **Geräte suchen** kann ein erneuter Suchlauf gestartet werden. Wenn dabei die Option **nur Unterschiede zum Projekt anzeigen** verfügbar und aktiviert ist, werden nur die Geräte angezeigt, die noch nicht im Gerätebaum im Projekt abgebildet sind.
- Über Schaltfläche **Ins Projekt kopieren** werden die im Fenster ausgewählten Geräte bzw. wenn nichts ausgewählt ist, alle gefundenen Geräte als Einträge in den Gerätebaum im Projekt übernommen. Abhängig vom Gerätetyp wird eventuell auch der Standarddialog zum Hinzufügen eines Geräts geöffnet, in dem dann das zu kopierende Gerät ausgewählt werden kann (z.B. bei Profibus).

Zusätzliche Funktionen für PROFINET IO Geräte:

- Zu jedem Gerät werden in weiteren Spalten im Fenster 'Verfügbare Geräte' auch **Stationsname**, **Stationsadresse**, **MAC Adresse**, **IP Adresse** und **Subnetzmaske** angezeigt.

Achtung: Jedes Gerät muss einen **Stationsnamen** erhalten, da dieser für bestimmte Funktionalitäten im Netzwerkbetrieb benötigt wird!

- Wenn beim Scannen der Filter **Nur unbenannte Stationen anzeigen** aktiviert ist, werden nur Geräte aufgelistet, für die noch kein Stationsname vergeben wurde.
- Der **Stationsname** kann hier im Dialog geändert werden (Editierfeld durch Doppelklick auf Feld) verändert werden. Um den neuen Namen im Bussystem bekannt zu machen, muss bei selektiertem Geräteeintrag die Schaltfläche **Taufen** betätigt werden. Daraufhin erscheint der Hinweis, dass der Bus abgeschaltet werden muss und eine neue Buskonfiguration erforderlich ist (durch Reset oder Download der Applikation), um die Änderung wirksam zu machen. Dies liegt in der Verantwortung des Anwenders! Der im Dialog angezeigte Stationsname wird auch im Programmiersystem im Gerätebaum verwendet.
- Auch die Einträge für Gerätename, IP Adresse und Subnetzmaske können auf diese Weise bereits hier **im Dialog bearbeitet** bzw. neu erstellt werden, bevor das Gerät mit diesen Einstellungen in die Konfiguration im Projekt übernommen wird. Wenn das Gerät übernommen wird, ohne Adressen zu definiert haben, werden diejenigen Voreinstellungen angewendet, die im PNIO Master Parameter Dialog des Geräteeditors für solche Fälle definiert sind. Die endgültige Definition der Adressen kann dann immer noch für jedes Gerät in dessen Konfigurationsdialog (PNIO Identifizierung) vorgenommen werden.

- Um ein im Fenster 'Verfügbare Geräte' aufgelistetes Gerät an der Hardware selbst zu identifizieren, selektieren Sie den Eintrag im Dialog und betätigen Schaltfläche **Identifizieren**. Das Gerät sollte darauf mit einem Blinksignal reagieren.
- Die Modul **Id Nummer** wird nur dann in der Liste angezeigt, wenn (1) sie vom Slave-Gerät bereitgestellt wird, (2) wenn Slave und Master Teil desselben logischen Netzwerks sind und (3) wenn IP-Adresse und Netzwerkmaske des Slaves korrekt gesetzt sind. Wenn die ID-Nummer nicht vorliegt, kann ein Slave-Gerät dennoch ins Projekt "kopiert" werden, wenn es durch die sonst verfügbaren Informationen eindeutig identifiziert werden kann.
- Beachten Sie die Möglichkeit die Konfigurationseinstellungen eines PROFINET IO Slave-Geräts mit denen des entsprechenden Hardware-Geräts zu abzugleichen.

Zusätzliche Funktionen für Profibus DP Geräte:

Der Profibus Scan zeigt eine Liste der gefundenen Profibus Slave Geräte an. Zu jedem Slave wird die Stationsadresse, die Ident-Nummer, der Status (Vergleich mit projektiertem Slave) und die vom Slave gemeldete IO-Konfiguration angezeigt. Die Zuordnung von gescanntem und projektiertem Slave findet über die Stationsadresse statt. Wird kein projektiertes Gerät mit der selben Stationsadresse gefunden, ist der Status „Neu“. Wenn die Zuordnung zu einem projektiertem Gerät möglich ist wird zusätzlich die Modul-Konfiguration verglichen. Bei Abweichungen wird ein rotes Fehler-Icon mit entsprechender Meldung angezeigt, sonst ist der Status „Ok“. Mit dem Ins Projekt kopieren Button können die gescannten Geräte ins Projekt übernommen werden.

Mittels der verfügbaren Informationen (Ident Nummer für Slaves, IO-Config für Module) ist keine eindeutige Zuordnung zu den Geräten eines bestimmten Herstellers möglich, daher öffnet sich jeweils der „Gerät anhängen...“ Dialog, in welchem die gescannten Geräte den im Projekt vorhandenen Geräten manuell zugeordnet werden können.

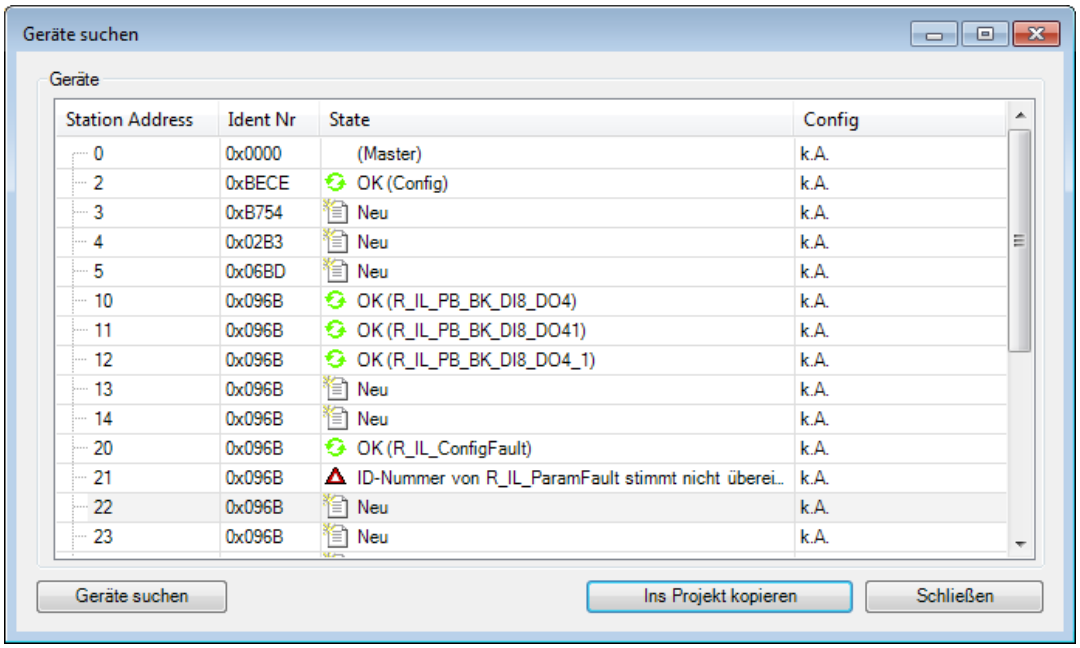


Abb. 100 Dialog "Scan Devices" für Profibus DP

Zusätzliche Funktionen für EtherCAT Geräte:

Wenn Sie ein EtherCAT Master Modul eingefügt haben und den Befehl Geräte suchen ausführen, wird Ihnen eine Liste aller verfügbaren EtherCAT Slaves angezeigt. Wie beim 'Geräte suchen' für andere Feldbusgeräte werden Ihnen fett gedruckte Einträge angezeigt, wenn es mehr als ein Gerät mit der gleichen Gerätebeschreibung gibt. Durch einen Doppelclick auf diesen Eintrag öffnet sich eine Liste und das gewünschte Gerät kann ausgewählt werden.

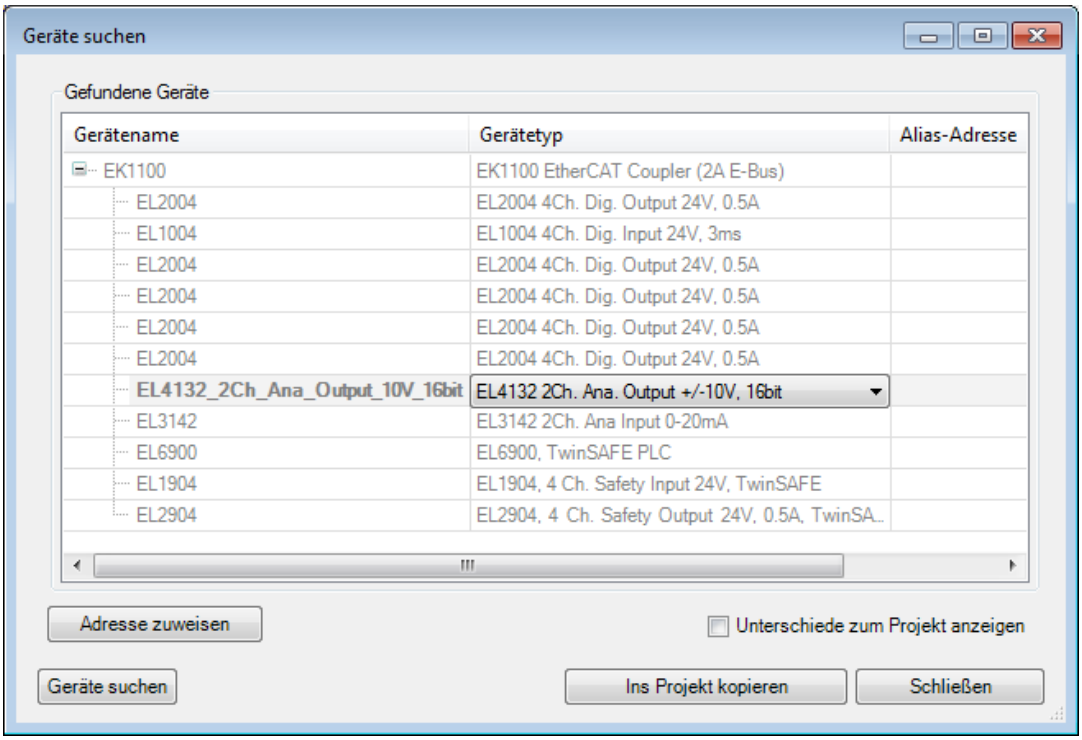


Abb. 101 Dialog "Scan Devices" für EtherCAT

Konfigurationsabgleich

Beachten Sie, dass diese Funktionalität nur für PROFINET IO Bussysteme verfügbar ist!

Der Befehl (Kategorie Geräte) bietet einen komfortablen Weg, die Konfigurationseinstellungen eines PROFINET Slave-Geräts, die im Projekt definiert sind, mit denen des entsprechenden Geräts (gleiche Geräte-ID und Hersteller-ID) in der angebundenen Hardware abzugleichen. Dies bedeutet, dass die lokalen Einstellungen mit den Hardware-Einstellungen überschrieben werden.

Folgende Schritte sind dazu auszuführen: Die Kommunikationseinstellungen entsprechend setzen. Dann im Offline-Modus das betreffende Geräteobjekt im Gerätebaum auswählen und Befehl "Konfigurationsabgleich" aus dem Kontextmenü auswählen. Der Dialog "**Geräte suchen**" wird geöffnet und die Suche nach passenden Geräten in der Hardware wird gestartet. Die gefundenen werden aufgelistet. Diejenigen, die bereits im Projekt verwendet werden (identifiziert anhand des Stationsnamens), können mit **Verwendete Geräte ausblenden** unsichtbar gesetzt werden. Der Suchprozess kann über die Schaltfläche **Geräte suchen** erneut gestartet werden.

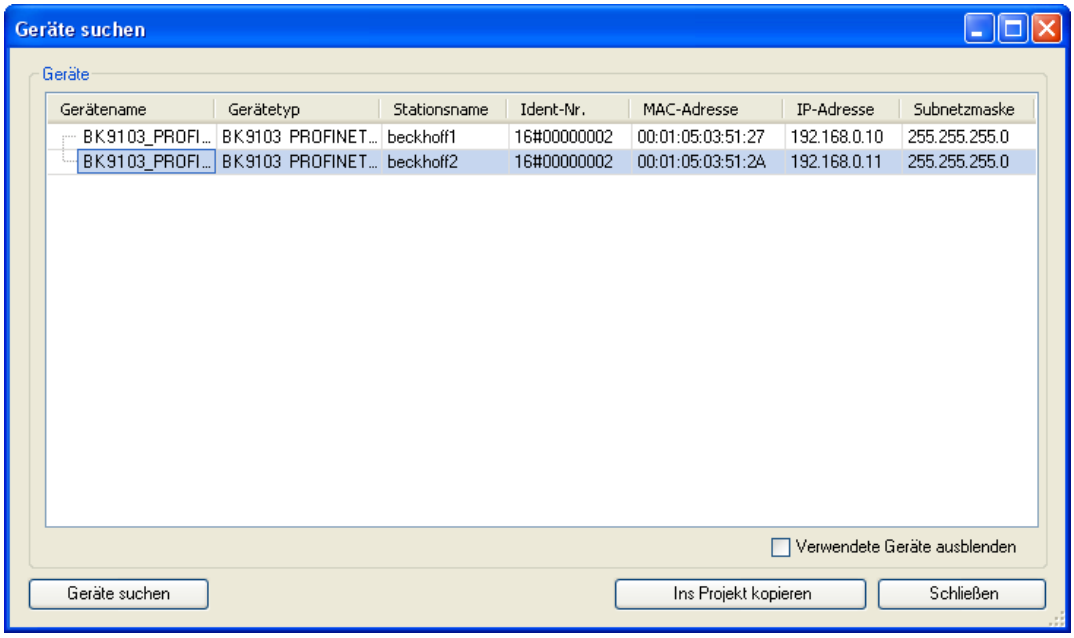


Abb. 102 Dialog "Geräte suchen" für Offline-Konfigurationsabgleich, Beispiel

Um die Konfigurationseinstellungen des Geräts im Projekt mit denen des Hardware-Geräts zu überschreiben, wähle man das gewünschte Gerät in der Liste aus und betätigen Schaltfläche **Ins Projekt kopieren**. Möglicherweise vorliegende Sub-Geräteobjekte werden da durch nicht entfernt.

12.11.3 Geräte-Repository

Befehle der Kategorie Geräte - Geräte-Repository

Die Funktionen zum Verwalten von Gerätedefinitionen auf dem lokalen System und in Projekten werden vom 'Device Repository' Plug-In bereitgestellt. Es liefert u.a. Befehle der Befehlskategorie 'Geräte', die standardmäßig im 'Tools' Menü zu finden sind.

Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

Die Befehle:

- Geräte-Repository...

Hinweis: Standardmäßig ist auch das Plug-In 'DeviceObject Plug-In' installiert, das Dialoge und Funktionen zur Handhabung der Geräte im Projekt bereitstellt.

Geräte-Repository...

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Dieser Befehl (Kategorie 'Geräte') öffnet den Dialog 'Geräte-Repository'. Ein Geräte-Repository ist eine Datenbank für Gerätebeschreibungen, die auf dem lokalen System installiert sind, um für das Programmieren in »PLC Designer« verfügbar zu sein. Das Installieren und Deinstallieren der Geräte wird im Geräte-Repository selbst vorgenommen:

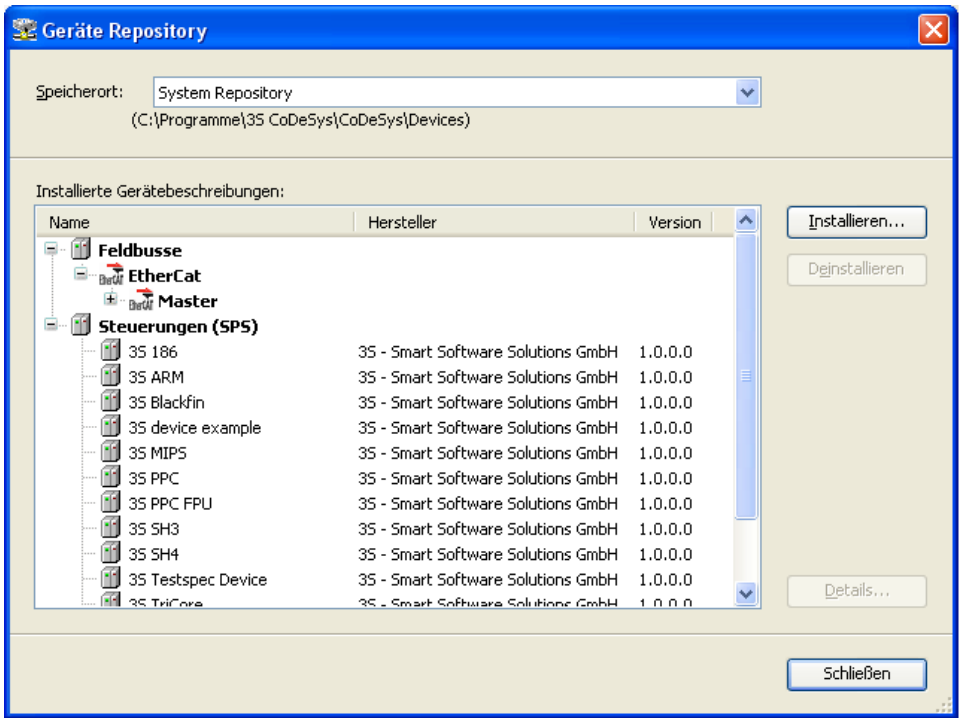


Abb. 103 Geräte-Repository Dialog

Speicherort: Geräte-Repositories können an verschiedenen Orten auf dem System angelegt sein. Die Auswahlliste zeigt die aktuell eingerichteten Speicherorte. Standardmäßig wird mit der Installation von »PLC Designer« zumindest das "System Repository" angelegt.

Installierte Gerätebeschreibungen: Die bereits installierten Geräte werden jeweils mit Angabe von **Name**, **Hersteller** und **Version** des Geräts angezeigt. Durch die Darstellung nach Gerätekategorien erhält die Liste eine Baumstruktur (siehe oben in Abbildung "Geräte-Repository Dialog" z.B. Kategorien "Feldbusse" und "Steuerungen"). Die "Äste" können mit Hilfe der Plus- und Minus-Zeichen geöffnet/geschlossen werden.

Details...: Diese Schaltfläche öffnet für das gerade in der Liste ausgewählte Gerät den "Details"-Dialog, der zusätzliche Information aus der Gerätebeschreibungsdatei darstellt: Gerätename, Hersteller, Kategorien, Version, Bestellnummer, Beschreibung.

Installieren...: Verwenden Sie diese Schaltfläche um ein Gerät neu auf dem lokalen System zu installieren, so dass es im Programmiersystem verfügbar wird. Der Dialog 'Installiere Gerätebeschreibung' wird geöffnet, wo man das System nach der entsprechenden Gerätebeschreibungsdatei durchsuchen kann. Für die Standard-Geräte ist hierfür der Dateityp-Filter auf "*.**devdesc.xml**"(Device Description) zu stellen. Aber auch herstellerspezifische Beschreibungsdateien, wie z.B. "gsd"-Dateien für Profibus DP Module, eds- und dcf-Dateien für CAN-Geräte, können ausgewählt werden.
Sobald die Auswahl mit OK bestätigt wird, schließt der Dialog und das neue Gerät wird im 'Geräte- Repository'-Dialog im Gerätebaum eingefügt.

Hinweis: Während der Installation werden die Gerätebeschreibungsdateien und alle zusätzlichen Dateien, die von dieser referenziert werden, an einen internen Speicherort kopiert. Somit haben Änderungen an den Originaldateien keinen Effekt mehr auf die installierten Geräte. Um solche Änderungen wirksam werden zu lassen, müssen die Geräte neu installiert werden. Es wird empfohlen, die interne Versionsnummer einer Gerätebeschreibung nach einer Modifikation zu verändern.

Achtung: Das interne Geräte-Repository darf nicht manuell verändert werden. Kopieren Sie keine Dateien ins Repository oder von dort woanders hin! Verwenden Sie immer den Geräte-Repository-Dialog um Geräte zu installieren oder zu deinstallieren!

Deinstallieren: Dieser Befehl dient zur Deinstallation des aktuell in der Liste ausgewählten Gerätes. Nach Betätigen einer Sicherheitsabfrage, ob der Befehl auch wirklich ausgeführt werden soll, wird das Gerät aus dem Geräte-Repository gelöscht und ist nicht mehr für die Verwendung im Programmiersystem verfügbar.

Die aktuelle Liste der installierten Geräte wird angeboten, wenn ein Geräte-Objekt über den Gerät anhängen- oder Objekt hinzufügen-Dialog im Geräte-Fenster eingefügt wird.

12.12 Gerätekommunikation

Die Befehlskategorie 'Gerätekommunikation' enthält Befehle zum Arbeiten im Dialog "Kommunikationseinstellungen" des generischen Geräte-Editors.

12.12.1 Befehle der Kategorie Gerätekommunikation

Die Befehle der Kategorie "Gerätekommunikation" sind standardmäßig im Kontextmenü verfügbar, wenn im Dialog Kommunikationseinstellungen des Geräteeditors gearbeitet wird. Dieser Dialog dient dazu, die Kommunikation über Gateway zum Zielgerät aufzusetzen. Für einige der Befehle gibt es auch eine Schaltfläche im Dialog.

Verfügbare Befehle:

- Gerät hinzufügen...
- Gateway hinzufügen...
- Gerätenamen ändern...
- Zum lokalen Gateway verbinden...
- Ausgewähltes Gerät löschen...
- Gateway bearbeiten...
- Suche nach Gerät über Name
- Suche nach Gerät über Adresse
- Netzwerk durchsuchen...
- Aktiven Pfad setzen

12.12.2 Gerät hinzufügen

Symbol:

Dieser Befehl (Kategorie Gerätekommunikation), für den es auch eine Schaltfläche im Dialog Kommunikationseinstellungen gibt, öffnet den Dialog **Gerät hinzufügen**, wo manuell ein Gerät definiert werden kann, das unterhalb des gerade ausgewählten Gateway-Eintrags eingefügt werden soll (Siehe hierzu auch die Funktionalität Netzwerk durchsuchen).

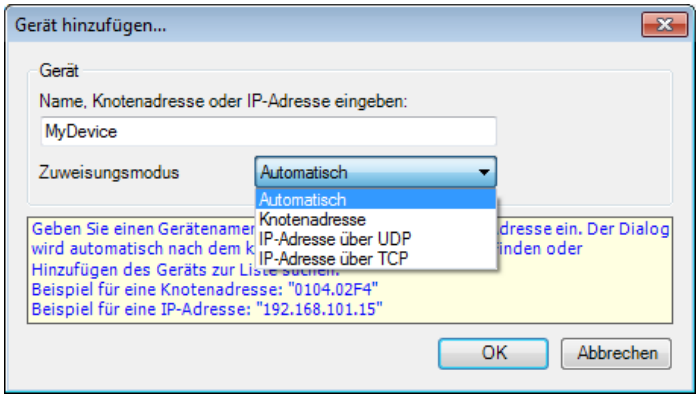


Abb. 104 Dialog "Gerät hinzufügen"

Hier kann das neue Gerät mit seinem **Namen** (Beispiel:"mynewdevice"), seiner **Knotenadresse** (Beispiel: "0104.02F4") oder **IP-Adresse** (Beispiel "192.168.101.15") angegeben werden.

Abhängig davon, ob ein Name oder eine Adresse angegeben werden, muss der entsprechende **Zuweisungsmodus** gewählt werden, d. h. wie das System vorgehen soll, um das gewünschte Gerät einzufügen. In den meisten Fällen wird Ihre Angabe eindeutig sein, so dass Modus "Automatisch" funktionieren wird. Die anderen beiden Modi werden nur benötigt, falls die Geräteangabe genauer spezifiziert werden muss, weil sie nicht offensichtlich als Name bzw. Adresse erkannt werden kann. Beispiel: Sie möchten ein Gerät hinzufügen, das Sie mit seiner Knotenadresse "AFFE" angeben. "AFFE" könnte aber auch der Gerätename sein, also muss Modus "Knotenadresse" eingestellt werden, damit die Angabe entsprechend behandelt wird.

- Automatisch: Der Dialog sucht automatisch den passenden Mechanismus um das Gerät zu finden und der Liste hinzuzufügen
- Knotenadresse: Wenn diese Option gesetzt ist, muß eine eindeutige Knotenadresse angegeben werden (Beispiel: "02F4"). Das Netzwerk wird daraufhin nach einem Gerät mit dieser Adresse durchsucht und wenn es gefunden wird, wird dieses der Liste hinzugefügt.
- IP-Adresse über UDP: Wenn diese Option gesetzt ist, muß eine eindeutige IP-Adresse angegeben werden (Beispiel: "192.168.101.15"). Das Netzwerk wird daraufhin nach einem Gerät mit dieser Adresse durchsucht, um dessen Knotenadresse zu ermitteln und es zur Liste hinzuzufügen. Optional kann der UDP Port für das Gerät angehängt werden (Beispiel: "192.168.192.15:1740").
- IP-Adresse über TCP: Wenn diese Option gesetzt ist, muß eine eindeutige IP Adresse angegeben werden (Beispiel: "192.168.101.15"). Das Netzwerk wird daraufhin nach einem Gerät mit dieser Adresse durchsucht, um dessen Knotenadresse zu ermitteln und es zur Liste hinzuzufügen. Optional kann der TCP Port für das Gerät angehängt werden (Beispiel: "192.168.192.15:11740").

Wenn ein Gerät angegeben wird, das es im Netzwerk nicht gibt, wird dennoch ein Geräteeintrag mit einem Default-Namen ("NewDevice") bzw. einer Default-Adresse ("FFFF.FFFF.FFFF") in der Liste eingefügt. Der Befehl "Adresse auflösen" kann verwendet werden, um das Gerät später, wenn es verbunden ist, zu identifizieren. Beachten Sie, dass ein Netzwerk-Scan das Gerät von der Liste entfernt, wenn es zum Zeitpunkt der Suche nicht verfügbar ist.

12.12.3 Gateway hinzufügen...

Symbol:

Dieser Befehl (Kategorie Gerätekommunikation), für den es auch eine Schaltfläche im Dialog Kommunikationseinstellungen gibt, öffnet den **Gateway** Dialog, wo ein Gateway definiert und zur aktuellen Konfiguration hinzugefügt werden kann. Geben Sie einen symbolischen **Namen**, den **Treiber**-Typ und treiberspezifische Einstellungsparameter (**IP-Adresse** und **Port**) für den Gateway ein. Um die Parameterwerte zu bearbeiten, öffnen Sie mit einem Doppelklick auf die jeweilige Spalte ein Eingabefeld. Nach dem Schließen des Dialogs mit OK wird dern neue Gateway-Eintrag im Konfigurationsbaum des Dialogs Kommunikationseinstellungen hinzugefügt.

Dieser Dialog kann verwendet werden, um die Gateway-Definition später zu bearbeiten (siehe Gateway bearbeiten...).

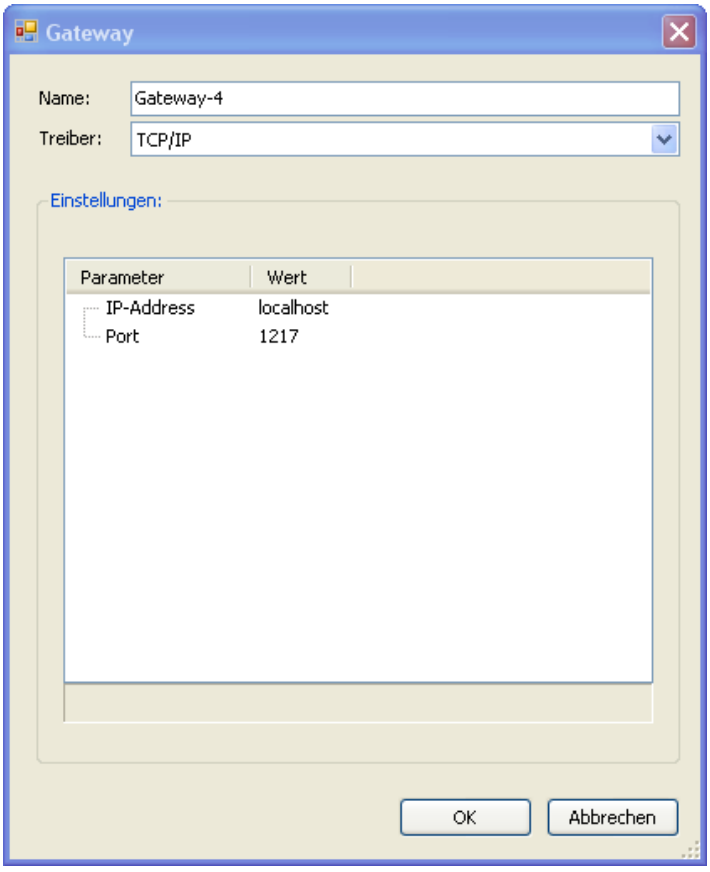


Abb. 105 Gateway Dialog

12.12.4 Gerätenamen ändern...

Dieser Befehl (Kategorie Gerätekommunikation) öffnet den gleichnamigen Dialog zum Ändern des (**Aktuell**en) Gerätenamens wie er im Konfigurationsbaum in den Kommunikationseinstellungen verwendet wird. Der neue Name kann manuell im Feld hinter **Neu:** eingegeben werden, oder die Standardoption "<Gerätename>" (Name des Geräts wie im Gerätebaum des Projekts verwendet) kann von der Auswahlliste genommen werden.

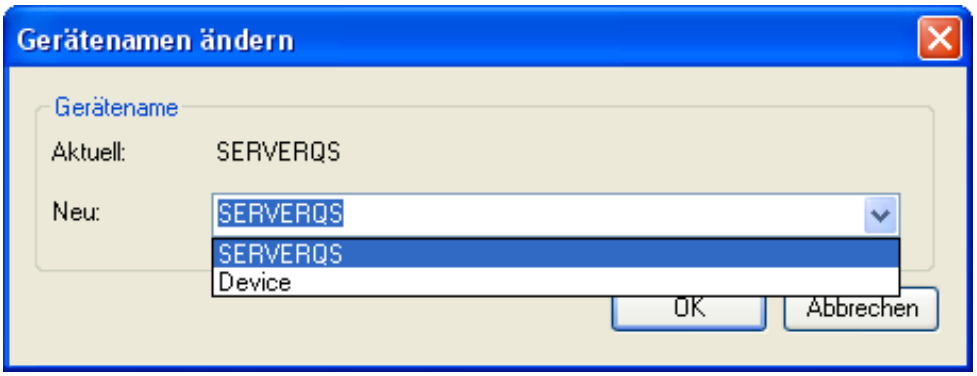


Abb. 106 Dialog "Gerätenamen ändern"

12.12.5 Zum lokalen Gateway verbinden...

Dieser Befehl (Kategorie Gerätekommunikation), für den es auch eine Schaltfläche im Dialog Kommunikationseinstellungen gibt, öffnet einen Dialog für die Konfiguration eines lokalen Gateways und zur manuellen Bearbeitung der Datei "Gateway.cfg"

Achtung: Eine saubere Konfiguration des Gateways erfordert detaillierte Kenntnisse. Im Zweifelsfall sind die Standardeinstellungen der manuellen Konfiguration vorzuziehen!

Router:

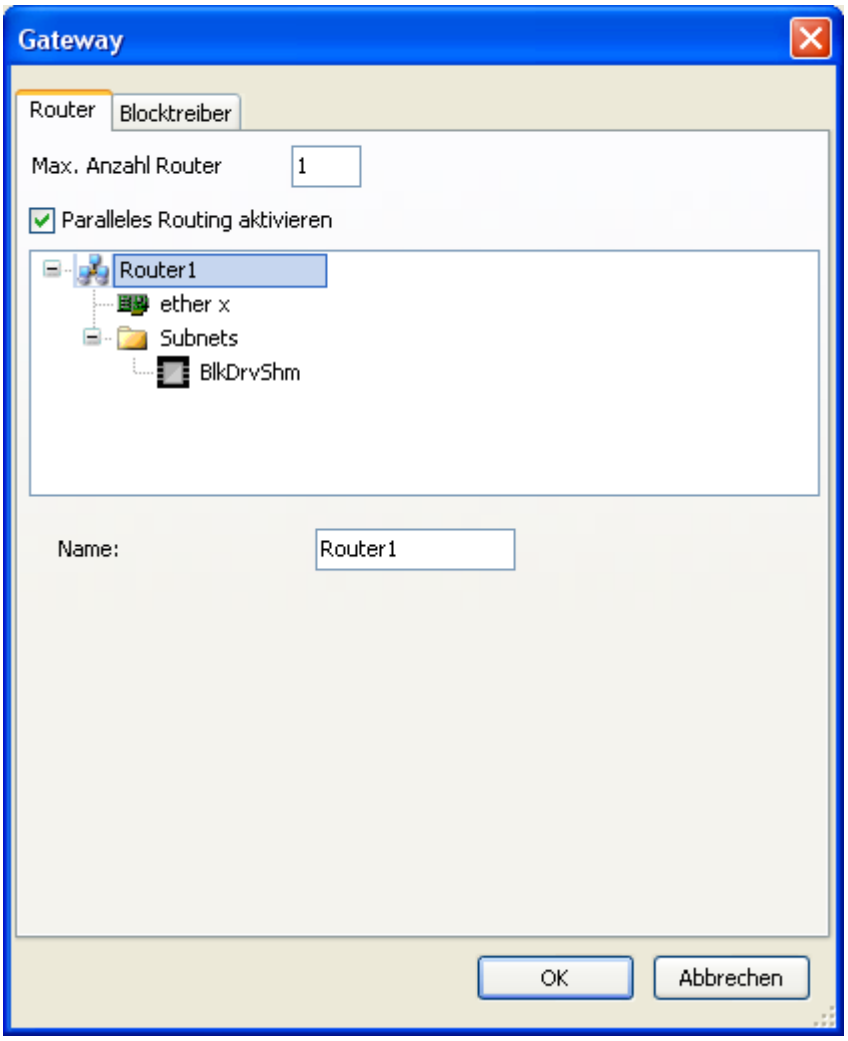
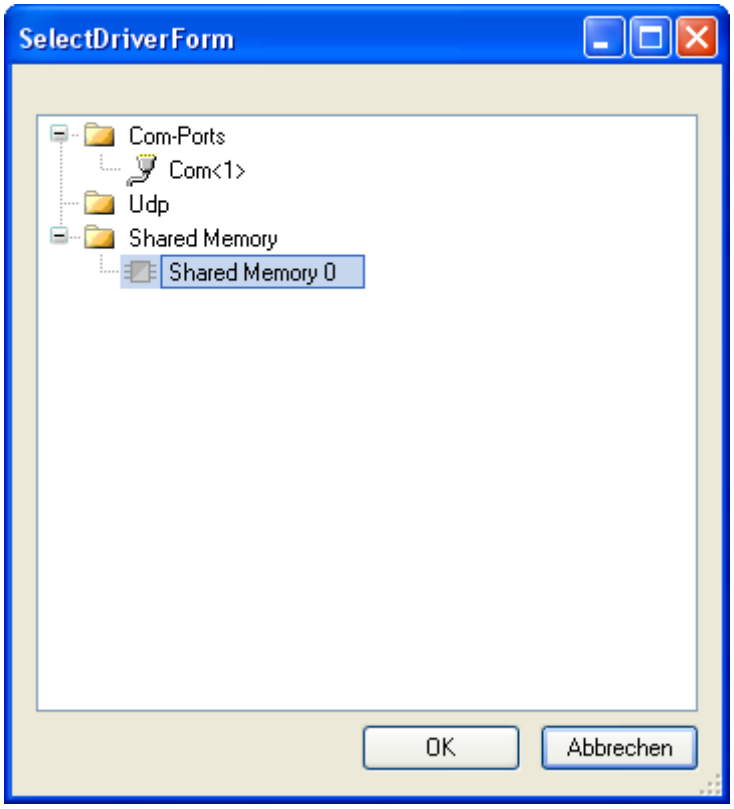


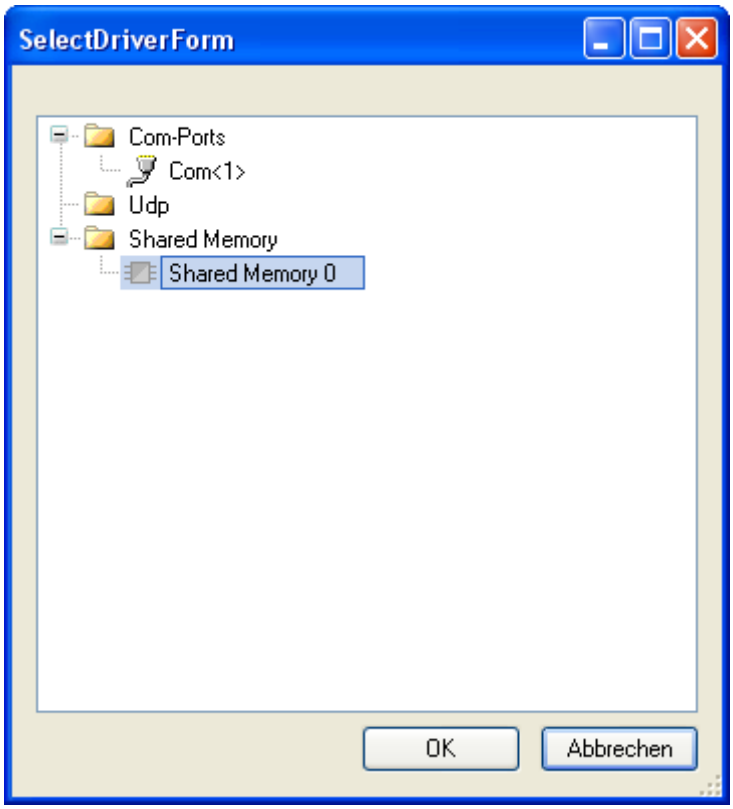
Abb. 107 Dialog Gateway - Router

Im Eingabefeld rechts von **Max. Anzahl Router** kann eine obere Grenze für die Anzahl an Routern angegeben werden. Die Aktivierung des Kontrollkästchens **Paralleles Routing** aktivieren erlaubt die direkte Kommunikation zweier Netzwerke, die einen gemeinsamen Knoten besitzen, welcher jedoch in jedem Netzwerk eigenständig repräsentiert und mit einer eigenen Adresse belegt ist. Beim Senden einer Nachricht von dem einem Netzwerk zum anderen über diesen gemeinsamen Knoten wird die Adressierung beim Durchlaufen dieses Knotens von derjenigen im ersten auf die Adressierung im zweiten Netzwerk abgebildet.

In Fenster unterhalb werden alle Router zusammen mit ihren Netzwerken und Schnittstellen dargestellt. Grundsätzliche ist jede Schnittstelle einem eigenen **Router** unterstellt, welcher selbst Teil des **Hauptnetzes** ist. Um Ressourcen zu sparen ist es jedoch möglich einem Router (bis zu 15) weitere Schnittstellen hinzuzufügen, jedoch nicht auf gleicher Stufe, sondern innerhalb von Subnetzen, die im Ordner **subnets** angezeigt werden.

Jeder Eintrag der Liste kann durch Anklicken ausgewählt werden. Das zugehörige Kontextmenü enthält eine Auswahl aus folgenden Kommandos:

Hinzufügen	Dient dazu ein neues Subnetz zum ausgewählten Ordner hinzuzufügen. Öffnet dazu einen Dialog, welcher alle bisher konfigurierten Blocktreiber zur Auswahl stellt:  Wählen Sie einen Eintrag durch Anklicken aus und bestätigen Sie Ihre Auswahl mit OK.
Löschen	Dient dazu den ausgewählten Router mit all seinen angehängten Komponenten zu löschen
Neu	Dient dazu einen neuen Router einzutragen
Entfernen	Dient dazu die ausgewählte Schnittstelle zu löschen

Hauptnetz setzen	Dient dazu die Schnittstelle des Hauptnetzes zu setzen. Bei Aufruf des Kommandos öffnet ein Dialog, welcher alle bisher konfigurierten Blocktreiber zur Auswahl stellt:  Wählen Sie einen Eintrag durch Anklicken aus und bestätigen Sie Ihre Auswahl mit OK.

Der untere Abschnitt enthält die Eigenschaften des ausgewählten Eintrags. Im Fall eines Treibers entsprechen diese gerade der im Reiter Blocktreiber vorgenommenen Konfiguration. Im Fall eines Routers kann hier sein **Name** editiert werden.

Blocktreiber:

Das obere Fenster enthält die Ordner Com-Ports, Udp und Shared Memory, in denen alle Blocktreiber, die Teil des Gateways sind, gemäß ihrem Typ gesammelt sind. Ein Eintrag kann durch Anklicken ausgewählt werden. Das zugehörige Kontextmenü enthält eines der folgenden Kommandos:

Löschen	um den ausgewählten Blocktreiber zu löschen
Neu	um einen Com-Port zum Ordner Com-Ports hinzuzufügen
Neu	um einen Udp Blocktreiber zum Ordner Udp hinzuzufügen
Neu	um Shared memory zum Ordner Shared Memory hinzuzufügen

Hinweis: Es wird ausdrücklich empfohlen den Inhalt des Ordners Shared Memory nicht zu verändern!

Achtung: Wenn in der Konfiguration zum ersten Mal ein Blocktreiber für einen Com-Port hinzugefügt wird, muss die Konfigurationsdatei **gateway.cfg** manuell erweitert werden. Öffnen Sie dazu die Datei, die standardmäßig im Verzeichnis GatewayPLC des »PLC Designer« Installationsverzeichnisses (z.B. C:\Programme\Lenze\PLCDesigner\GatewayPLC) liegt, in einem Texteditor und fügen in Sektion [ComponentManager] folgenden Eintrag hinzu: Component.xy=CmpBlkDrvSimpleCom, wobei xy zur fortlaufenden Komponentennummerierung passen muss. Wenn dieser Eintrag fehlt, wird der Blocktreiber beim Starten des Gateways nicht geladen!

Im unteren Teil des Dialogs werden die Konfigurationseinstellungen des gerade ausgewählten Eintrags angezeigt

Im Fall eines Com-Ports kann sein Name, sein Subindex innerhalb des Gateways (COM Port) und seine Baudrate editiert werden. Bei Aktivierung des Kontrollkästchens Autoadressierung aktivieren werden doppelt vergebene Adressen automatisch aufgelöst. Sofern eindeutig und nicht anders spezifiziert im Feld Lokale Adresse, stimmt die Standardeinstellung für die lokale Adresse gerade mit dem Eintrag in COM Port überein.

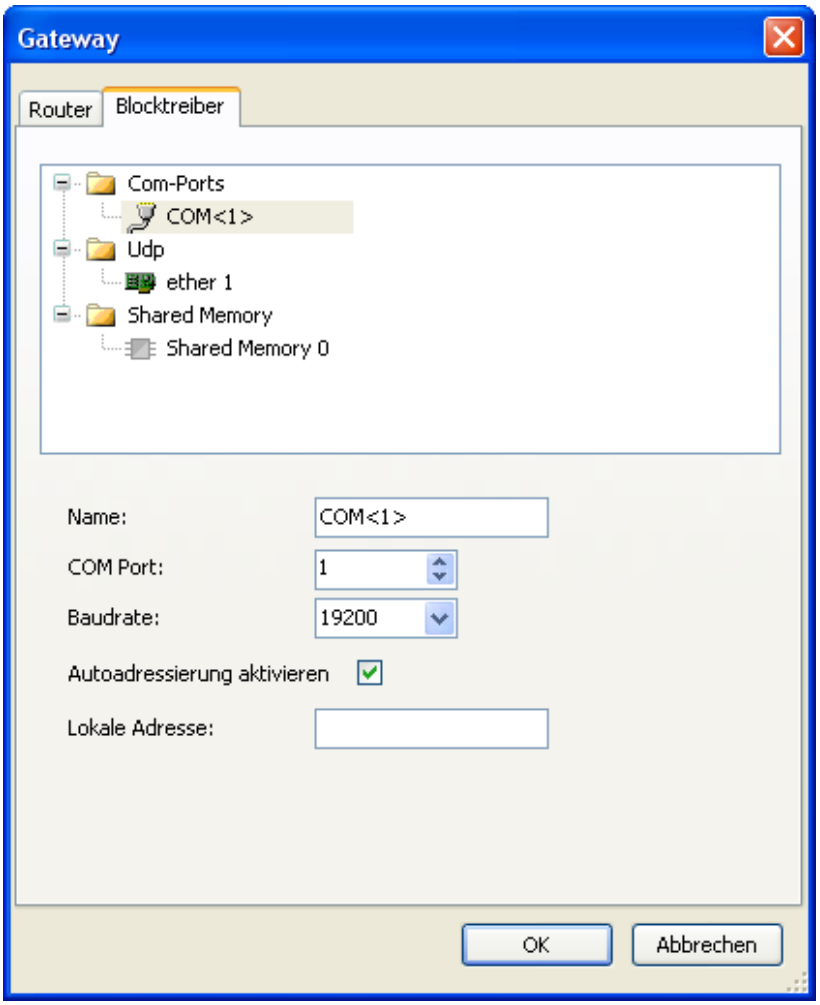


Abb. 108 Dialog Gateway - Blocktreiber (Com Port)

Im Fall eines Udp Blocktreibers kann der **Name** editiert und eine spezielle **IP Adresse** angegeben werden, falls diese von der des standardmäßig gewählten Routers abweichen sollte. Weiterhin können **Netzwerkmaske** und der **Port Index** angegeben werden (zum Auswählen der Port-Nummer des »PLC Designer« Gateways; der Port Index reicht von 0 bis 3, wobei Index 0 der Port-Nummer 1740 entspricht, und Index 3 der Port-Nummer 1743). Bei Eingabe einer **Ppp Remote Adresse** wird eine logische Punkt-zuPunkt Verbindung zwischen der Udp Schnittstelle und einem durch diese Adresse bestimmten Knoten aufgebaut. In Folge wird die Udp Schnittstelle ausschließlich mit diesem Knoten kommunizieren und im Netzwerk werden keine Broadcasts mehr versendet werden.

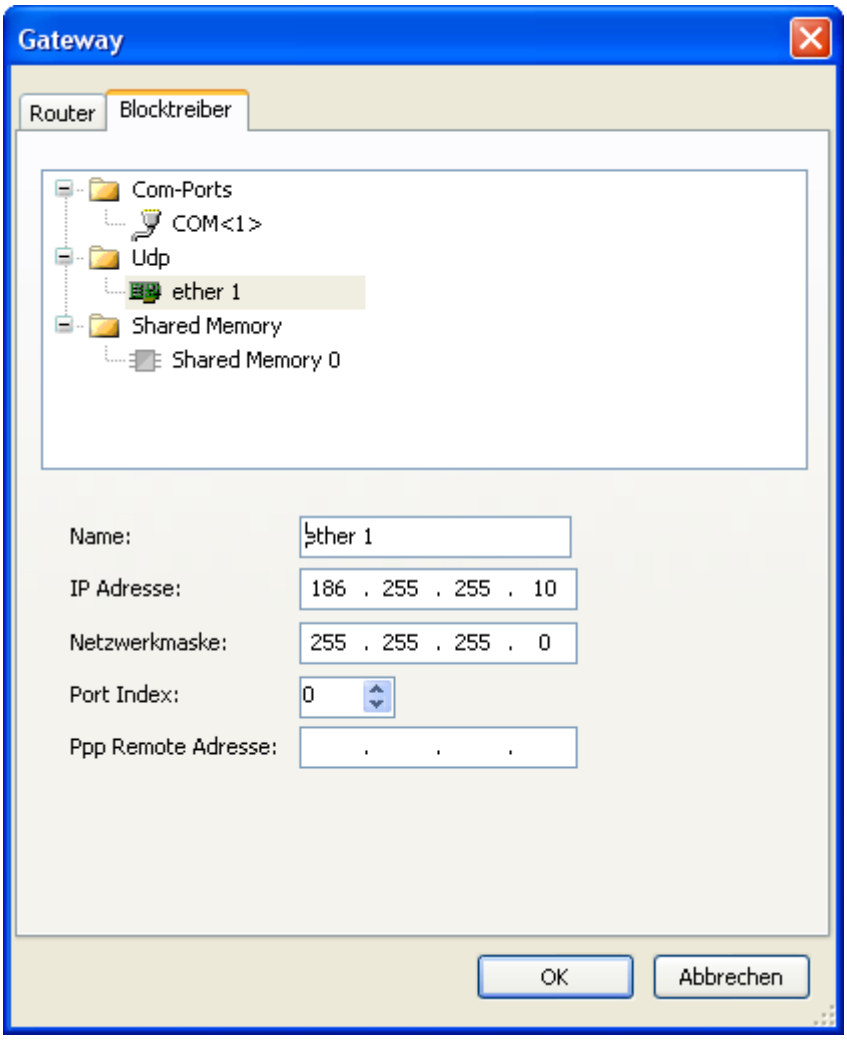


Abb. 109 Dialog Gateway - Blocktreiber (Udp)

Im Fall von Shared Memory kann der **Name** eingegeben werden. Sofern Sie die lokale Adresse des Shared Memory festsetzen wollen, spezifizieren Sie diese im Eingabefeld **Forced Address**.

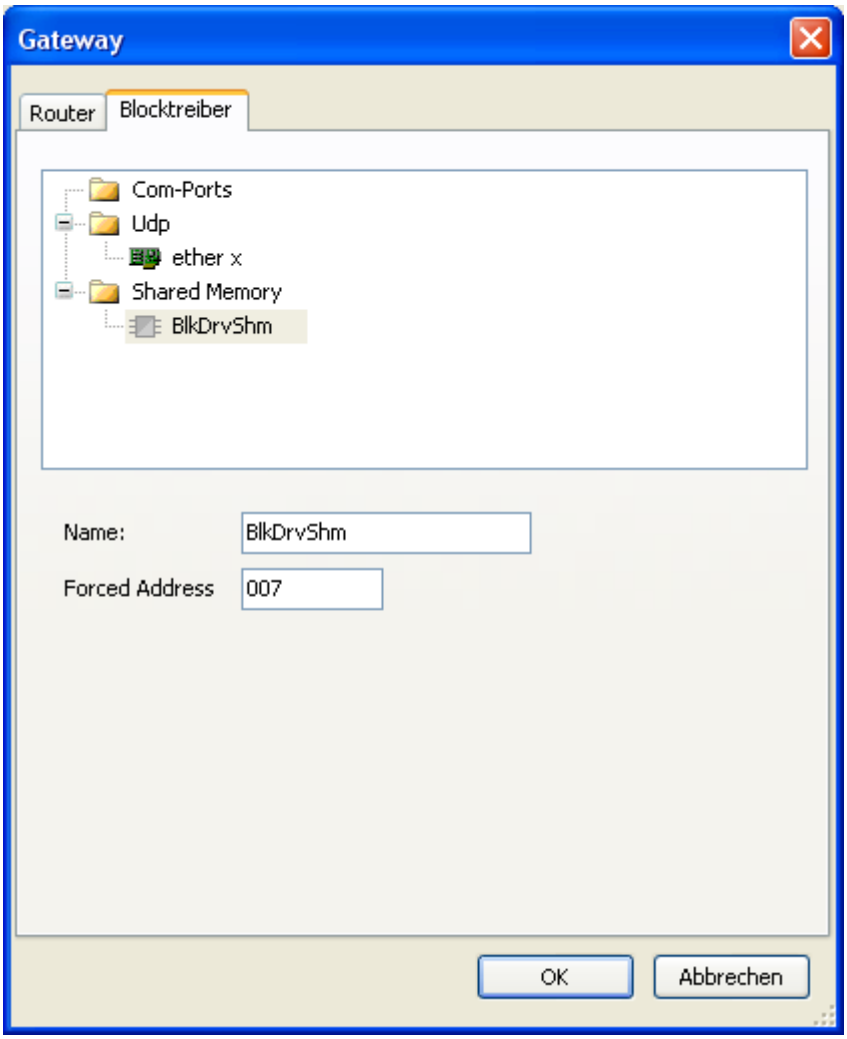


Abb. 110 Dialog Gateway - Blocktreiber (Shared Memory)

Achtung: Nach Beenden der Konfiguration im Gateway-Dialog (Schließen mit OK) und einer ev. Änderung der gateway.cfg-Datei muss der Gateway gestoppt und neu gestartet werden. Ansonsten werden die Änderungen nicht wirksam!

12.12.6 Ausgewähltes Gerät löschen...

Dieser Befehl (Kategorie Gerätekommunikation) entfernt das gerade ausgewählte Gerät von der Geräteliste im Dialog Kommunikationseinstellungen.

12.12.7 Gateway bearbeiten...

Dieser Befehl (Kategorie Gerätekommunikation) öffnet den Gateway Dialog, der in den Kommunikationseinstellungen für das Hinzufügen eines Gateways bzw. das Ändern der Einstellungen des gerade ausgewählten Gateways bereitsteht.

12.12.8 Suche nach Gerät über Name

Symbol:

Dieser Befehl (Kategorie Gerätekommunikation) dient dazu, das Netzwerk nach Geräten zu durchsuchen, die denselben Namen tragen wie im Dialog Kommunikationseinstellungen in der Geräteliste angegeben (Beachtung der Groß/Kleinschreibung!). Wenn welche gefunden werden, wird jedes davon mit seiner Knotenadresse der Liste hinzugefügt, d.h. der Eintrag wird mit Name + Knotenadresse dargestellt.

Beispiel

USER1 [057B]

12.12.9 Suche nach Gerät über Adresse

Symbol:

Dieser Befehl (Kategorie Gerätekommunikation) dient dazu, das Netzwerk nach Geräten zu durchsuchen, die denselben Kontenadresse haben wie im Dialog Kommunikationseinstellungen in der Geräteliste angegeben . Wenn welche gefunden werden, wird jedes davon mit seinem Namen der Liste hinzugefügt, d. h. der Eintrag wird mit Name + Knotenadresse dargestellt.

Beispiel

USER1 [057B]

12.12.10 Netzwerk durchsuchen...

Symbol:

Dieser Befehl (Kategorie Gerätekommunikation), für den es auch eine Schaltfläche im Dialog Kommunikationseinstellungen gibt, startet die Suche nach verfügbaren Geräten im lokalen Netzwerk und der Konfigurationsbaum wird entsprechend aktualisiert. Der Netzwerk-Scan wird auch durch einen Doppelklick auf den Gateway-Eintrag aufgerufen bzw. er wird automatisch durchgeführt, wenn man versucht einzuloggen obwohl noch kein Pfad zur Steuerung definiert ist (siehe hierzu auch: Einloggen).

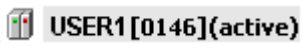
Die gefundenen Geräte werden unterhalb desjenigen Gateway-Eintrags aufgelistet, unterhalb dem gerade ein Eintrage selektiert war, als der Befehl aufgerufen wurde.

12.12.11 Aktiven Pfad setzen

Symbol:

Dieser Befehl (Kategorie Gerätekommunikation), für den es auch eine Schaltfläche im Dialog Kommunikationseinstellungen gibt, setzt den Kommunikationskanal, der gerade im Baum unterhalb des Gateways ausgewählt ist, als aktiven Pfad zur Steuerung. Das heißt, dass alle Kommunikationsaktionen sich direkt auf diesen Kanal beziehen. Der Befehl wird auch durch einen Doppelklick auf den Kanaleintrag aufgerufen.

Der gerade aktive Pfad wird in der Liste fett dargestellt und "(aktiv)" wird angehängt:



Wenn ein Gerät als aktiver Pfad gesetzt ist, aber beim letzten Netzwerk-Scan nicht gefunden wurde, wird es in Kursivschrift dargestellt.

Hinweis: Die Information, welches Gerät zuletzt als "aktiver Pfad" eingestellt war, wird lokal auf dem System gespeichert und kann beim Einloggen mit nicht konfigurierten Kommunikationseinstellungen automatisch wieder angewendet werden.

12.13 AWL/FUP/KOP

Die Befehlskategorie 'AWL/FUP/KOP' enthält Befehle zum Arbeiten im kombinierten FUP/KOP/AWL-Editor.

12.13.1 FUP/KOP/AWL Editor Befehle

Das Plug-In 'Network List Editor' stellt Befehle der Kategorie 'FUP/KOP/AWL' für das Arbeiten im FUP/KOP/AWL-Editor bereit, dem gemeinsamen Editor für alle drei Sprachen FUP (Funktionsplan), KOP (Kontaktplan) und AWL (Anweisungsliste):

Standardmäßig sind diese Befehle im 'FUP/KOP/AWL'-Menü zu finden, das bereitsteht, wenn der Editor geöffnet ist. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog verändert werden.

Hinweis: Für das Einfügen von Elementen in der FUP und KOP Editoransicht beachten Sie bitte auch die Möglichkeit die Elemente direkt mit Drag&Drop aus der Werkzeugbox oder von einer anderen Position innerhalb des Editors zu holen.

12.13.2 Netzwerk einfügen

Symbol:

Tastaturkürzel: <Strg> + <I>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt ein Netzwerk im FUP/KOP/IL-Editor ein.

Wenn der Cursor gerade in einem schon vorhandenen Netzwerk steht, wird das neue Netzwerk **oberhalb** von diesem eingefügt. Wenn der Cursor gerade zwar im Editorfenster, aber nicht in einem Netzwerk steht, wird das neue Netzwerk am Ende der bestehenden Netzwerkliste angehängt. Die Netzwerknummerierung wird automatisch aktualisiert.

Beachten Sie, dass im Unterschied zu »PLC Designer« V2.3 auch im AWL-Editor Netzwerke als Struktureinheit verwendet werden.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.3 Netzwerk einfügen unterhalb

Symbol:

Tastaturkürzel: <Strg> + <T>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt ein Netzwerk im FUP/KOP/IL-Editor ein. Wenn der Cursor gerade in einem schon vorhandenen Netzwerk steht, wird das neue Netzwerk **unterhalb** von diesem eingefügt. Wenn der Cursor gerade zwar im Editorfenster, aber nicht in einem Netzwerk steht, wird das neue Netzwerk am Ende der bestehenden Netzwerkliste angehängt. Die Netzwerknummerierung wird automatisch aktualisiert.

Beachten Sie, dass im Unterschied zu »PLC Designer« V2.3 auch im AWL-Editor Netzwerke als Struktureinheit verwendet werden.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.4 Kommentierung ein/aus

Symbol:

Tastaturkürzel: <Strg> + <O>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') kann im FUP/KOP/AWL-Editor verwendet werden, um ein Netzwerk "auszukommentieren" bzw. es wieder in den normalen Status zurückzubringen. Er wirkt auf das Netzwerk, in dem gerade der Cursor steht.

Ein "auskommentiertes" Netzwerk wird wie ein Kommentar dargestellt und behandelt, d.h. das Netzwerk wird nicht abgearbeitet.

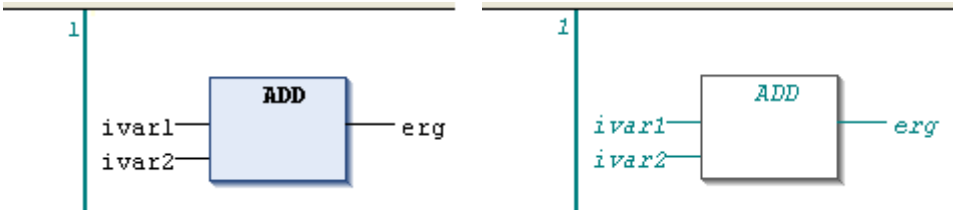


Abb. 111 Beispiel: Netzwerk im "Normal-Status" und im "Kommentar-Status"

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.5 Zuweisung einfügen

Symbol:

Tastaturkürzel: <Strg> + <A>

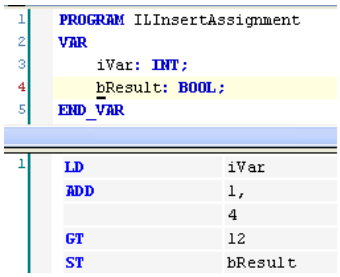
Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt eine Zuweisung im FUP oder KOP Editor ein. In der AWL-Ansicht ist er nicht verfügbar.

Abhängig von der aktuellen Cursor-Position wird die Zuweisung direkt vor einem ausgewählten Eingang (Cursor Position 2), direkt nach einem ausgewählten Ausgang (Cursor Position 4) oder wenn das gesamte Netzwerk oder Subnetzwerk markiert ist, an dessen Ende eingefügt. (Cursor Position 6 bzw. 11).

Im FUP Editor wird eine Zuweisung als Linie mit nachfolgenden drei Fragezeichen "???" eingefügt, in KOP als Spule und "???":

Um eine Zuweisung zu definieren, wählen Sie den Platzhaltertext "???" aus und ersetzen ihn mit dem Namen der Variablen, der das von links kommende Signal (Wert) zugewiesen werden soll. Die Eingabehilfe steht dabei zur Verfügung.

In AWL wird eine Zuweisung über die Operatoren "LD" und "ST" programmiert. Beispiel:



```
PROGRAM ILInsertAssignment
VAR
    iVar: INT;
    bResult: BOOL;
END_VAR
```

```
LD      iVar
ADD     1,
        4
GT      12
ST      bResult
```

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.6 Bausteinaufruf einfügen

Symbol:

Tastaturkürzel: <Strg> + <B>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') wird verwendet, um in einem FUP-, KOP- oder AWL-Netzwerk ein Baustein-Element einzufügen. Dabei kann es sich um den Aufruf eines Operatorbausteins, eines Programms, eines Funktionsblocks, einer Funktion oder einer Schnittstelle handeln. Im AWL Editor werden zu diesem Zweck die entsprechenden Anweisungen eingefügt.

Wenn der Befehl gewählt wird, öffnet der Dialog der Eingabehilfe, der die verfügbaren Bausteinkategorien anbietet. Wenn Sie einen Baustein ausgewählt und mit OK bestätigt haben, wird eine entsprechende Box mit Ein- und Ausgängen bzw. die entsprechenden AWL-Anweisungen an der aktuellen Cursor-Position im Netzwerk eingefügt.

Alternativ kann Befehl Leeren Baustein einfügen verwendet und der gewünschte Bausteintyp direkt angegeben werden, oder es können Standardbausteine aus dem Werkzeuge-Fenster mit drag&drop direkt ins Editorfenster gezogen werden. Zum "Drag&Drop" von Elementen siehe: Arbeiten im FUP- und KOP-Editor.

Sehen Sie im folgenden die editor-spezifischen Besonderheiten beim Einfügen eines Bausteinaufrufelements:

FUP / KOP:

- Bausteine vom Typ **Programm** oder **Funktionsblock** werden immer in Serie eingefügt, das heißt die Abarbeitungslinie wird mit dem obersten Eingang und dem obersten Ausgang des eingefügten Bausteins verbunden.

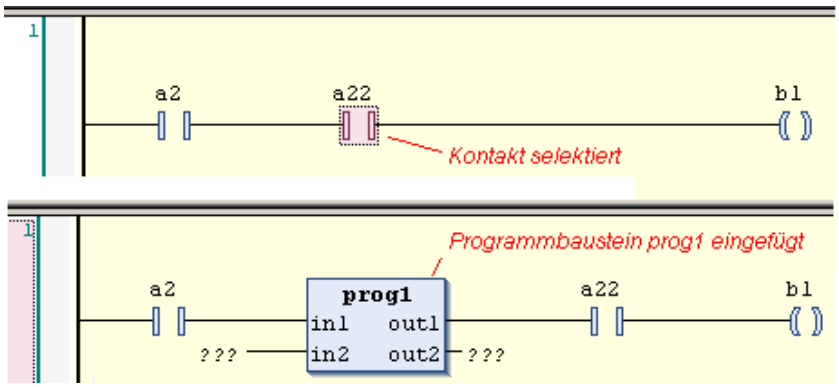


Abb. 112 Beispiel: Einfügen eines Funktionsblocks in KOP

- Der Text im eingefügten Bausteinelement gibt den **Bausteintyp** an (z.B. F_TRIG) und ist editierbar. Durch Ersetzen dieses Textes mit dem Typnamen eines anderen gültigen Bausteins kann der Bausteinaufruf durch einen anderen **ersetzt** werden. Ersetzt werden kann auch dadurch, dass das vorhandene Bausteinelement selektiert wird und ein anderes an dieser Stelle eingefügt wird. Dabei ist zu beachten, dass **bereits definierte Belegungen** von Ein- und Ausgängen eines Bausteins gemäß ihrer vorigen Anordnung von oben nach unten beibehalten werden, außer der neue Baustein hat eine kleinere Höchstzahl an Eingängen. Dann werden die nicht mehr verknüpfbaren Ein- und Ausgänge von unten her gelöscht.
- Innerhalb von Parallelschaltungen in einem KOP Netzwerk erhält man keine Einfügepositionen für einen Baustein aus der Werkzeuge-Box. Hintergrund: Ein Bausteinaufruf braucht eine direkte Verbindung zur Stromleitung (linke und rechte vertikale Linie der KOP-Netzwerkliste).

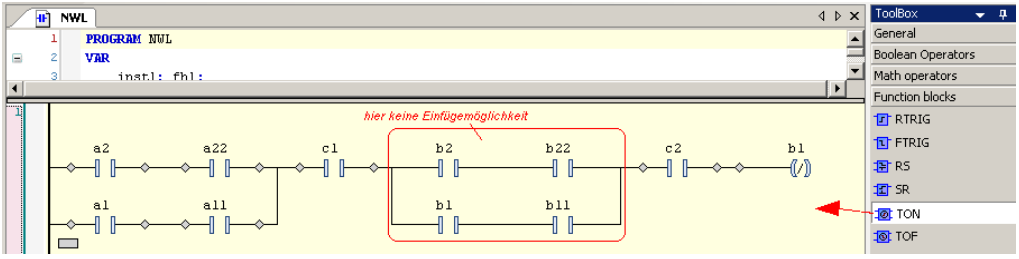


Abb. 113 Beispiel: Einfügepositionen für Bausteinaufruf in KOP-Netzwerk

- Aufrufe von bestimmten **Operatoren**-Bausteinen werden im KOP Editor automatisch mit **EN/ENO** bzw. nur EN Ein- und Ausgängen eingefügt. Mit EN und ENO versehen werden dabei solche, deren Ausgang nicht vom Typ BOOL ist (z.B. ADD, SEL, BOOL_TO_INT), nur mit EN versehen werden solche, deren Ausgang vom Typ BOOL ist (z.B. EQ, GE, GT). Bei Bausteinen mit ENO-Ausgang können an den anderen Ausgängen des Bausteins nicht weitere Bausteinaufrufe eingefügt werden.
- **Funktionsblock**-Bausteine haben ein editierbares Textfeld oberhalb der Box, in dem die lokale Instanzvariable eingetragen werden muss. Wenn ein solches Funktionsblock-Instanz.-Element ersetzt wird, indem ein anderer Funktionsblock-Typ eingefügt wird, muss auch die Instanzdefinition erneut eingegeben werden.
- In den Boxen von Funktions- und Funktionsblockaufrufen werden die **formalen Namen der Ein- und Ausgänge** angezeigt. Der Ausgang einer Funktion (Rückgabewert) wird allerdings ohne Namen dargestellt.
- Falls sich die **Schnittstelle eines Bausteins** geändert hat (z.B. veränderte Anzahl von Ausgängen), können die Bausteinparameter mit dem Befehl Parameter aktualisieren aktualisiert werden. Achtung, diese **Aktualisierung erfolgt nicht automatisch** wie in »PLC Designer« V2.3!

- Einfügepositionen: Ein neuer Bausteinaufruf wird folgendermaßen an der aktuellen Cursor-Position eingefügt:
  - Wenn gerade ein Eingang selektiert ist (Cursor-Position 2), wird die Bausteinbox vor diesem eingefügt und mit seinem ersten Ein- und ggfs. Ausgang im bestehenden Leitungszweig angeschlossen.
  - Wenn gerade ein Ausgang selektiert ist (Cursor-Position 4), wird die Bausteinbox nach diesem eingefügt und mit seinem ersten Ein- und ggfs. Ausgang im bestehenden Leitungszweig angeschlossen.
  - Wenn gerade eine Bausteinbox selektiert ist (Cursor-Position 3), wird dieser Baustein durch den neuen ersetzt. Soweit möglich bleiben die Belegungen der Ein- und Ausgänge in ihrer Anordnung von oben nach unten erhalten. Wenn der alte Baustein mehr Eingänge oder Ausgänge hatte als der neue, werden die nicht verknüpfbaren (von unten her) gelöscht.
  - Wenn ein Sprung oder Return-Element selektiert ist (Cursor-Position 3), wird die Bausteinbox davor eingefügt und mit seinem ersten Ein- und ggfs. Ausgang im bestehenden Leitungszweig angeschlossen.
  - Wenn ein gesamtes Netzwerk oder ein Sub-Netzwerk selektiert ist, (Cursor-Position 6 bzw. 11), wird die Bausteinbox nach dem letzten Element eingefügt und mit seinem ersten Eingang angeschlossen. Der Ausgang bleibt ohne Verbindung.
- Alle Baustein-Eingänge, die beim Einfügen nicht automatisch belegt werden, erhalten den Text "???". Dieser muss selektiert und mit dem Namen einer Variablen oder mit einer Konstanten ersetzt werden.
- Wenn ein Baustein am Ende eines Netzwerks eingefügt wird, bleibt sein Ausgang unverknüpft.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

AWL:

In AWL kann ebenfalls der Befehl 'Bausteinaufruf einfügen' verwendet werden, um einen Baustein aufzurufen. Wenn die Eingabehilfe zur Auswahl verwendet wird, und dort die Option 'Mit Argumenten einfügen' aktiviert ist, wird eine entsprechende CAL Anweisung gleich mit den entsprechenden Ein- und Ausgangsparametern eingefügt:

Beispiel:

```
1   CAL            TONinst(
              IN:= ???,
              PT:= ???,
              ET=> ???)
```

Abb. 114 Bausteinaufruf in AWL

Der Baustein "TON" wurde mit der Eingabehilfe ausgewählt; Die Eingabeparameter werden automatisch in den folgenden Zeilen vorgegeben und können nun definiert werden. Im gezeigten Bild wurden die "???" in der CAL-Zeile vom Anwender bereits durch "TONinst" (lokale Instanz des TON) ersetzt.

12.13.7 Leeren Baustein einfügen

Symbol:

Shortcut: <Ctrl> + <Shift> + <B>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') wird verwendet, um in einem FUP-, KOP- oder AWL-Netzwerk ein leeres Baustein-Element einzufügen.

Siehe Bausteinaufruf einfügen für ausführliche Informationen zum Einfügen von Bausteinen im FUP/KOP/AWL Editor. Im Unterschied zu Befehl Bausteinaufruf einfügen wird dabei die Eingabehilfe nicht automatisch geöffnet, sondern das Instanz-Feld oberhalb des Bausteinelements. Entscheiden Sie dann, welcher Bausteintyp benötigt wird:

- Wenn es ein Funktionsbaustein sein soll, geben Sie den gewünschten Instanzvariablennamen ein und schließen das Feld mit <Eingabe>. Die Eingabehilfe kann verwendet werden, um den Namen einer bereits existierenden Instanzvariablen auszuwählen. Nach der Eingabe einer bereits deklarierten Instanzvariable wird der Baustein entsprechend dargestellt. Wenn ein noch nicht bekannter Instanzname eingegeben wurde, muss auch noch der Name des gewünschten Funktionsbausteins angegeben werden. Dazu wechselt nach Schließen des Eingabefelds der Fokus automatisch auf das Eingabefeld innerhalb der Box gesetzt. In diesem Fall bietet die Eingabehilfe auch nur Funktionsbausteine an.

- Wenn der Baustein einen Operator, ein Program, eine Funktion oder eine Schnittstelle darstellen soll, drücken Sie, wenn der Mauszeiger im Instanzeingabefeld steht, die Nach-unten-Pfeiltaste. Der Eingabefokus wechselt dann zum Bausteintypenfeld (innerhalb des Bausteinelements), wo dann der gewünschte Operator-, Programm-, Funktions- oder Schnittstellenname eingegeben werden kann, direkt oder über die Eingabehilfe. Nach Beenden dieser Eingabe wird der Baustein entsprechend im Netzwerk dargestellt.

Im AWL-Editor werden die entsprechenden Anweisungen eingefügt.

12.13.8 Sprung einfügen

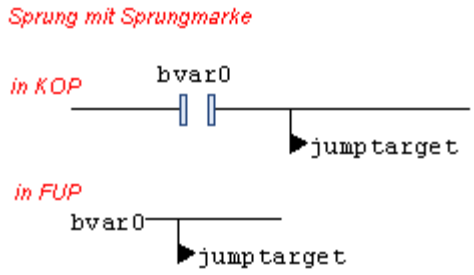
Symbol:

Tastaturkürzel: <Strg> + <L>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt einen Sprung ein. Das Sprungziel ist ein anderes Netzwerk, das hier durch seine Marke angegeben wird (Sprungmarke).

In **FUP** und **KOP**, wird das Sprung-Element abhängig von der aktuellen Cursor-Position, entweder direkt vor einem ausgewählten Eingang (Cursor Position 2), direkt nach einem ausgewählten Ausgang (Cursor Position 4) oder wenn das gesamte Netzwerk oder Subnetzwerk markiert ist, an dessen Ende eingefügt. (Cursor Position 6 bzw. 11).

Ein neues Sprungelement enthält den Text "???", der mit einem Mausklick ausgewählt werden und mit dem Namen der Marke des Netzwerks ersetzt werden kann, zu dem gesprungen werden soll.



In **AWL** wird ein Sprung mit Hilfe des JMP Operators programmiert. Beispiel:

```
PROGRAM IL
VAR
    bVar: BOOL;
    bVar2: BOOL;
END_VAR
```

```
LD          bVar
JUMPC       jumptarget
jumptarget:
LD          bVar2
```

Wird ein JMP Operator, der zuvor innerhalb des AWL-Editors ohne vorangehenden LD eingefügt wurde, in KOP konvertiert, so wird ein Dummy Operator "???" eingefügt.

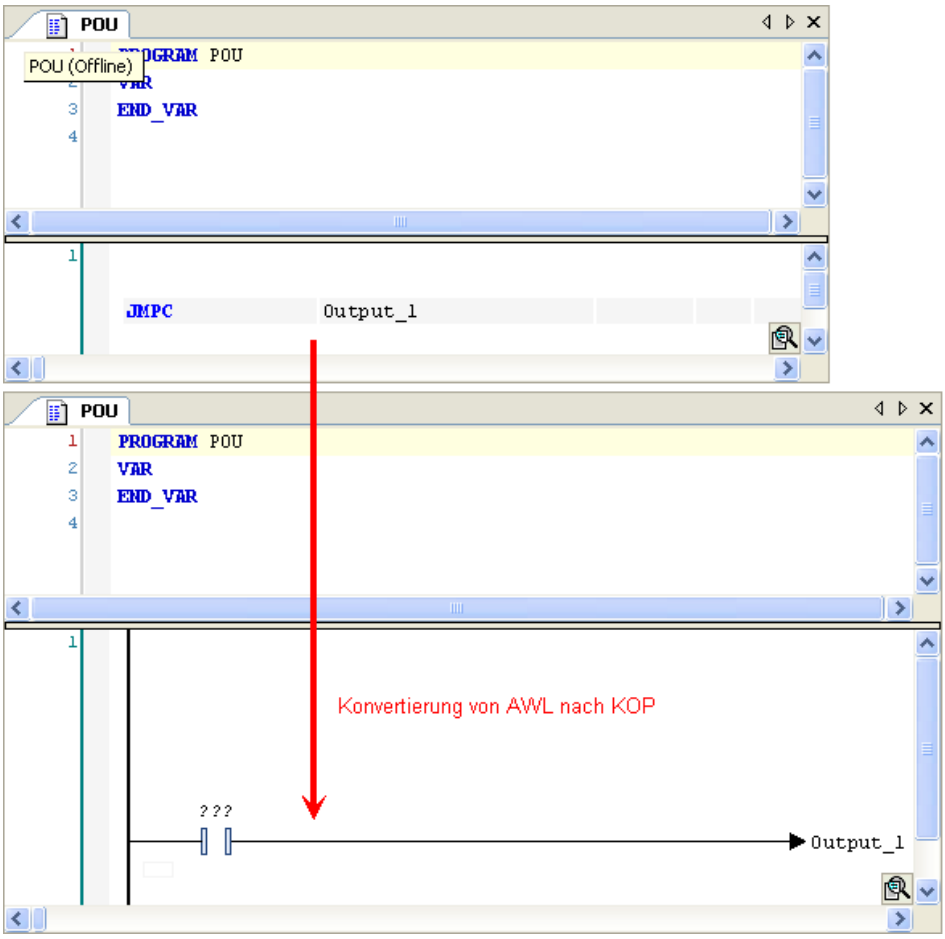


Abb. 115 Beispiel: Konvertierung eines JMP Operators

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.9 Sprungmarke einfügen

Symbol:

Das Kommando (Kategorie 'FUP/ KOP /AWL') fügt eine Sprungmarke in das aktuell ausgewählte Netzwerk ein. Die Sprungmarke kann bei einem Sprung als Ziel verwendet werden.

Bei Aufruf des Kommandos wird standardmäßig der Text "Label:" in das zugehörige Textfeld eingetragen; dieser darf beliebig abgeändert werden.

12.13.10 Return einfügen

Symbol:

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt eine RETURN-Anweisung an der aktuellen Cursor-Position ein.

FUP oder **KOP**:

Entsprechend der aktuellen Cursor-Position wird das Return-Element direkt vor einen ausgewählten Eingang (Cursor-Position 2), direkt nach einem ausgewählten Ausgang (Cursor-Position 4), direkt vor einer Verzweigung (Cursor-Position 5) oder am Ende des Netzwerks oder Subnetzwerks (Cursor-Position 6 oder 11).

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

AWL:

In AWL wird eine Anweisung mit Operator RETURN eingefügt.

12.13.11 Bausteineingang einfügen

Symbol:

Tastaturkürzel: <Strg> + <Q>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt einem bereits eingefügten erweiterbaren Baustein (AND, OR, ADD, MUL, SEL) im FUP- oder KOP-Editor einen weiteren Eingang hinzu. Der Befehl ist im AWL Editor nicht verfügbar.

Die maximale Anzahl von Eingängen hängt vom Bausteintyp ab (beispielsweise kann ADD 2 oder mehr Eingänge haben.

Um den Eingang an einer bestimmten Position bezüglich der schon vorhandenen einzufügen, selektieren Sie den Eingang, oberhalb dessen der neue eingefügt werden soll (Cursor-Position 1).

Um einem Baustein ganz unten einen weiteren Eingang hinzuzufügen, selektieren Sie den Rumpf des Bausteins.(Cursor-Position 3).

Der neue Eingang ist zunächst mit "???" belegt. Ersetzen Sie diesen Text mit dem Namen einer Variable oder mit einer Konstanten. Die Eingabehilfe steht dazu bereit.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.12 Spule einfügen

Symbol:

Tastaturkürzel: <Ctrl> + <A>

Dieses Kommando (Kategorie 'FUP / KOP / AWL') benutzt man um eine Spule parallel zu schon vorhandenen Spulen einzufügen.

Ist die markierte Position eine Verbindung zwischen Kontakten und Spulen, so wird die neue Spule als letzte eingefügt. Ist die markierte Stelle eine Spule, so wird die neue Spule parallel dazu direkt oberhalb eingefügt.

Die Spule trägt standardmäßig den Text "???". Zur Abänderung können Sie diesen Text anklicken und editieren oder aber den Eingabe-Assistenten benutzen.

Für weitere Optionen - Eingabe von Adressen, Zeilenumbrüchen bei Variablennamen und Kommentaren - beachten Sie bitte die Beschreibung des Dialogs Optionen: FUP/KOP/AWL-Editor.

12.13.13 Set-Spule einfügen

Symbol:

Dieses Kommando (Kategorie 'FUP / KOP / AWL') wird benutzt um eine Set-Spule einzufügen. Alternativ kann die Befehlskombination 'Spule einfügen' und' Extras' 'Set/Reset' verwendet werden, um eine solche Spule einzufügen.

12.13.14 Reset-Spule einfügen

Symbol:

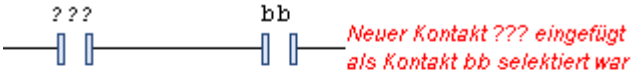
Dieser Befehl (Kategorie 'FUP/KOP/AWL') wird verwendet, um eine Reset-Spule einzufügen. Alternativ kann die Befehlskombination 'Spule einfügen' und' Extras' 'Set/Reset' verwendet werden, um eine solche Spule einzufügen.

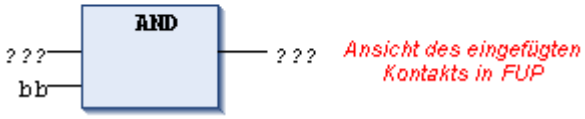
12.13.15 Kontakt einfügen

Symbol:

Tastaturkürzel: <Strg> + <K>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt in einem Kontaktplan-Netzwerk ein Kontakt-Element ein. Der Befehl ist in der FUP- und AWL-Ansicht des Editors nicht verfügbar, ein Kontakt wird beim Umschalten aber entsprechend konvertiert.

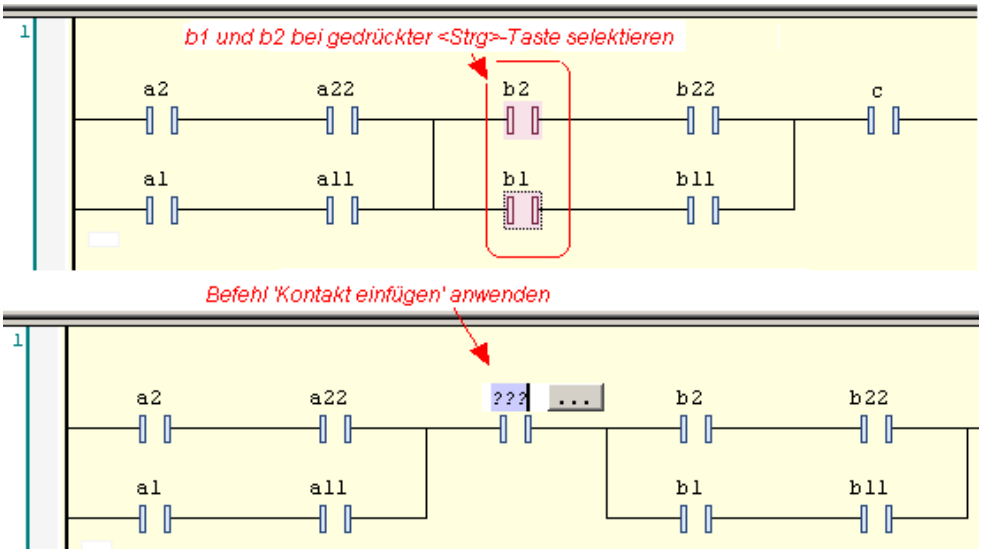




Der neue Kontakt wird an der aktuellen Cursor-Position in Reihe **vor** einem bestehenden Kontakt bzw. einer Bausteinbox eingefügt. Wenn die Cursor-Position im Ast einer bestehenden Parallelschaltung liegt, wird der neue Kontakt innerhalb dieser eingefügt.

Hinweis: Alternativ kann ein Kontakt-Element auch aus der Werkzeuge-Box oder von einer anderen Position im Editor geholt werden. Wenn es allerdings nicht innerhalb sondern vor, nach oder zwischen bereits bestehenden Parallelschaltungen eingefügt werden soll, ist dies nur über den Menübefehl möglich. Hierzu muss in jedem Zweig der Parallelschaltung eines der enthaltenen Kontakt-Elemente selektiert (Mehrfachselektion bei gedrückter <Strg>-Taste) und dann der Befehl angewendet werden.

Beispiel:



Beachten Sie auch die Befehle Kontakt einfügen (rechts), Parallelen Kontakt einfügen (oberhalb), Parallelen Kontakt einfügen (unterhalb).

Ein neues Kontaktelement enthält den Text "???", der mit einem Mausklick ausgewählt werden und durch den Namen oder der Adresse (abhängig von den aktuellen Einstellungen in den 'FUP, KOP und AWL Optionen') der gewünschten Variablen oder durch eine Konstante ersetzt werden kann. Dabei steht die Eingabehilfe zur Verfügung.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.16 Negierten Kontakt einfügen

Symbol:

Dieses Kommando (Kategorie 'FUP / KOP / AWL') wird benutzt um einen negierten Kontakt einzufügen. Alternativ kann eine Kombination der Kommandos 'Kontakt einfügen' und 'Negation' benutzt werden um einen negierten Kontakt einzufügen.

12.13.17 Kontakt einfügen rechts

Symbol:

Tastaturkürzel: <Strg> + <D>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt in einem Kontaktplan-Netzwerk ein Kontakt-Element ein. Es gilt dasselbe wie für den Befehl 'Kontakt einfügen, nur dass in diesem Fall das neue Element nicht links, sondern rechts von der aktuellen Cursor-Position eingefügt wird.

12.13.18 Kontakt parallel einfügen unterhalb

Symbol:

Tastaturkürzel: <Strg> + <R>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt in einem Kontaktplan-Netzwerk an der aktuellen Cursor-Position einen parallelen Kontakt ein. Es gilt dasselbe wie für Befehl 'Kontakt parallel einfügen oberhalb', nur dass der neue Kontakt **unterhalb** der ausgewählten Position erscheint.

12.13.19 Negierten Kontakt parallel einfügen (unterhalb)

Symbol:

Dieses Kommando (Kategorie 'FUP / KOP / AWL') wird benutzt um einen negierten Kontakt parallel einzufügen. Alternativ kann eine Kombination der Kommandos 'Kontakt parallel einfügen (unterhalb)' und 'Negation' benutzt werden um einen negierten Kontakt einzufügen.

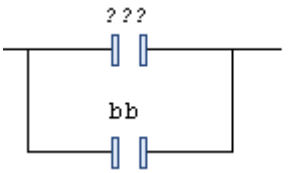
12.13.20 Kontakt parallel einfügen oberhalb

Symbol:

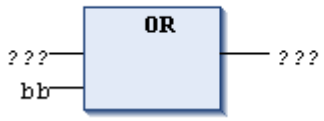
Tastaturkürzel: <Strg> + <P>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') fügt in einem Kontaktplan-Netzwerk **oberhalb** der aktuellen Cursor-Position einen parallelen Kontakt ein (Parallelschaltung). Dabei können auch mehrere bestehende Elemente selektiert werden, um einen neuen Kontakt parallel zu diesen zu legen. Der Befehl ist in der FUP- und AWL-Ansicht des Editors nicht verfügbar, ein Kontakt wird beim Umschalten aber entsprechend konvertiert.

Ein neues Kontaktelement enthält den Text "???", der mit einem Mausklick ausgewählt werden und durch den Namen oder der Adresse (abhängig von den aktuellen Einstellungen in den 'FUP, KOP und AWL Optionen') der gewünschten Variablen oder durch eine Konstante ersetzt werden kann. Dabei steht die Eingabehilfe zur Verfügung.



Einfügen eines parallelen Kontakts in KOP
(die selektierte Position war Kontakt "bb")



Ansicht des eingefügten parallelen Kontakts in FUP

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.21 Kontakte einfügen: Darunter einfügen

Tastaturkürzel: <Strg> + <F>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') ist nur im KOP Editor verfügbar. Er fügt die Elemente bzw. den Teil des Netzwerks, welcher vorher über einen "Kopieren" oder "Ausschneiden" Befehl in die Zwischenablage gebracht wurde, **unterhalb** der aktuellen Selektion ein. Dies entspricht dem allgemeinen "Einfügen" Befehl.

12.13.22 Kontakte einfügen: Rechts einfügen danach

Tastaturkürzel: <Strg> +<G>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') ist nur im KOP Editor verfügbar. Er fügt die Elemente bzw. den Teil des Netzwerks, welcher vorher über einen "Kopieren" oder "Ausschneiden" Befehl in die Zwischenablage gebracht wurde, **rechts** der aktuellen Selektion ein.

12.13.23 Kontakte einfügen: Darüber einfügen

Tastaturkürzel: <Strg> + <H>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') ist nur im KOP Editor verfügbar. Er fügt die Elemente bzw. den Teil des Netzwerks, welcher vorher über einen "Kopieren" oder "Ausschneiden" Befehl in die Zwischenablage gebracht wurde, **oberhalb** der aktuellen Selektion ein.

12.13.24 AWL-Zeile danach einfügen

Symbol:

Dieser Befehl (Kategorie 'AWL, FUP, KOP') ist nur im AWL-Editor verfügbar. Er fügt eine weitere Anweisungszeile unterhalb derjenigen ein, in der gerade der Cursor steht.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.25 AWL-Zeile löschen

Symbol:

Tastaturkürzel: <Strg> + <Del>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') ist nur im AWL-Editor verfügbar. Er löscht die Anweisungszeile, in der gerade der Cursor steht.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.26 Negation

Symbol: (FUP), (KOP)

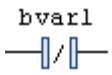
Tastaturkürzel: <Strg> + <N>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') schaltet den gerade selektierten Eingang Ausgang, Sprung oder die selektierte Return-Anweisung zwischen "negiert" und "nicht negiert" hin und her. Er ist im AWL-Editor nicht verfügbar. Dort müssen die entsprechenden Modifikatoren gesetzt werden.

Hinweis: Wenn zwischen FUP- bzw. KOP-Ansicht und AWL-Ansicht hin- und hergeschaltet wird, kann es vorkommen, dass die Negationen mancher Konstrukte zurückgesetzt werden, weil eine eindeutige Konvertierung nicht möglich ist.

Bei den Elementen Baustein, Sprung oder Return wird eine Negation durch ein kleines Kreissymbol an der betroffenen Ein- oder Ausgangsverbindung angezeigt.

Ein negierter Kontakt in KOP wird durch einen Schrägstrich im Kontaktsymbol angezeigt:



Um einen Kontakt oder eine Spule zu negieren, selektieren Sie das Element (Cursor-Position 8 bzw. 9) und wenden den Befehl an. Beachten Sie, dass die Werkzeuge-Box in Kategorie 'Kontaktplan Elemente' negierte Kontakt-Elemente zum Einfügen über drag&drop bereitstellt..

Um einen Eingang oder Ausgang zu negieren, selektieren Sie diesen an Cursor-Position 2 oder 4.

Um einen Sprung oder eine RETURN Anweisung zu negieren, selektieren Sie den letzen vorausgehenden Ausgang (Cursor-Position 4).

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.27 Flankenerkennung

Symbol: (FUP), (KOP)

Tastaturkürzel: <Strg> + <E>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') wird im FUP- oder KOP-Editor verwendet, um an einem boolschen Eingang einen Detektor für eine steigende oder eine fallende Flanke einzufügen. Dies entspricht der Verwendung der Bausteine R_TRIG (steigende Flanke: FALSE -> TRUE) bzw. F_TRIG (fallende Flanke TRUE -> FALSE).

Selektieren Sie den Eingang und führen den Befehl so oft aus, bis das gewünschte Symbol erscheint. Es wird gewechselt zwischen für steigende Flanke, für fallende Flanke und keinem Symbol

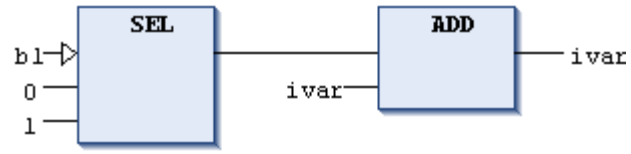


Abb. 116 Beispiel, Detektor für steigende Flanken am SEL-Operator

In diesem Beispiel wurde der Eingang bei b1 selektiert und der Befehl 'Flankenerkennung' ausgeführt. Der Detektor für steigende Flanken wurde eingefügt, d.h. der Ausgang von SEL liefert "1", wenn b1 von FALSE auf TRUE wechselt.

Der Befehl ist im AWL-Editor nicht verfügbar. FUP- oder KOP-Netzwerke, die Flankenerkennung enthalten, werden unverändert in die AWL-Ansicht übernommen.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.28 Set/Reset

Symbol:

Tastaturkürzel: <Strg> + <M>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') macht boolsche Ausgänge bzw. Spulen zu "Set" bzw. "Reset"-Ausgängen oder -Spulen. Er ist im AWL-Editor nicht verfügbar. Hier müssen die entsprechenden Modifikatioren gesetzt werden.

Wenn der Befehl an demselben Ausgang mehrfach hintereinander ausgeführt wird, wird nacheinander "Set", "Reset" und normaler Zustand eingestellt.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.29 Weiterverschaltung festlegen

Symbol:

Tastaturkürzel: <Strg> + <W>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') kann im FUP- und KOP-Editor für Baustein-Elemente verwendet werden, die mehrere Ausgänge haben, um festzulegen, welcher dieser Ausgänge mit der weiterführenden Abarbeitungslinie im Netzwerk verbunden werden soll.

Dies ist eine **erweiterte Möglichkeit gegenüber »PLC Designer« V2.3**, wo nur der obere Ausgang weiter verschaltet werden kann.

Beachten Sie die **Verschiebung der Belegungen** der Ausgänge, wenn die Weiterverschaltung umdefiniert wird; siehe unten in der Abbildung.

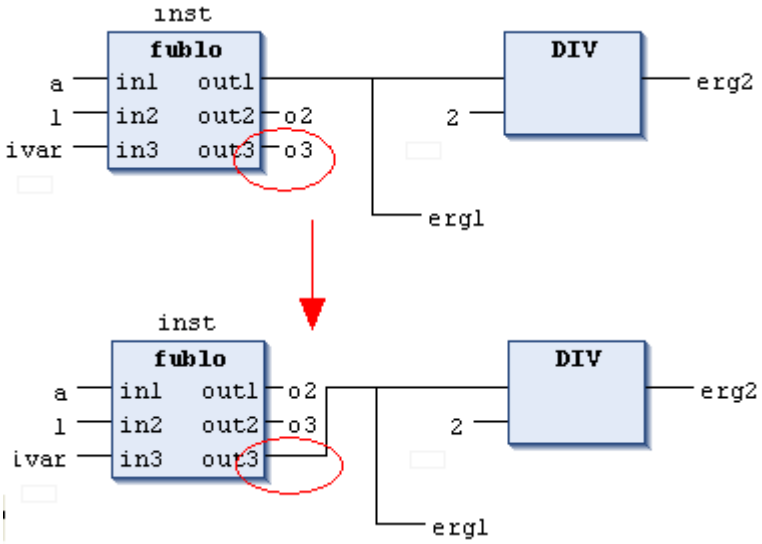


Abb. 117 Beispiel: Weiterverschaltung wird auf out3 gelegt:

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.30 Leitungsverzweigung einfügen

Symbol:

Tastaturkürzel: <Strg> + <V>

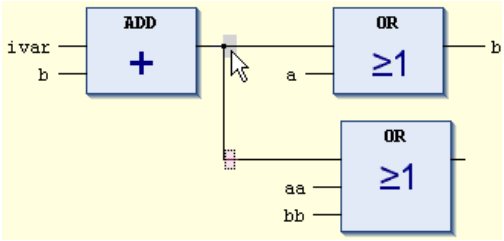
Dieser Befehl (Kategorie 'AWL, FUP, KOP') verzweigt die aktuelle Abarbeitungslinie in einem Netzwerk in FUP.

Die Abarbeitungslinie wird in zwei "Subnetzwerke" aufgespaltet: Eine weitere Abarbeitungslinie wird unterhalb der bestehenden eingefügt. Die beiden Subnetzwerke werden im Online-Modus entsprechend ihrer Position von oben nach unten abgearbeitet.

Wenn eine "Leitungsverzweigung" aus der Werkzeugbox oder von einer anderen Position im Editor hergezogen wird, werden die möglichen Einfügepositionen durch graue Positionsmarken angezeigt.

Eine Verzeweigung kann eingefügt werden an den Eingangskonnektoren von Bausteinen, die nicht innerhalb eines Subnetzwerks liegen, an Ausgangskonnektoren (Cursor-Position 4) eines Bausteins, wenn dieser nicht (auch nicht indirekt) mit dem Eingang eines anderen Bausteins im Subnetzwerk verbunden ist, an der Verbindung zwischen Kontakten und Spulen (Cursor-Position 10), oder an einem Kontakt (Cursor-Position 8). Eine Verzweigung kann nicht innerhalb einer mit "OR" versehenen Kontaktegruppe und innerhalb von Mehrfachzuweisungsgruppen eingefügt werden.

Jedes Subnetzwerk bekommt einen eigenen "Marker", ein aufrechtstehendes Rechteck-Symbol, das als mögliche Cursor-Position (11) zum Selektieren des Subnetzwerks dient.



Jedes Subnetzwerk kann wiederum verzweigt werden, was erlaubt, ein weitverzweigtes Netz und "Subnetzwerken" innerhalb des Haupt-Netzwerkes anzulegen. Zum Hinzufügen von Subnetzwerken zu Subnetzwerken siehe die Befehle: Leitungsverzweigung oberhalb einfügen und Leitungsverzweigung unterhalb einfügen.

Siehe Verzweigungen in FUP/KOP/AWL für mehr Informationen zu Verzweigungen und Subnetzwerken.

Verbindungen zwischen verschiedenen Verzweigungsarmen oder Subnetzwerken können mit Hilfe der Befehle Anschlüsse trennen und Anschlüsse verbinden entfernt oder hergestellt werden.

Der Befehl ist im AWL-Editor nicht verfügbar, aber beim Umschalten zwischen den Editoransichten wird entsprechend konvertiert.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.31 Leitungsverzweigung oberhalb einfügen

Symbol:

Dieser Befehl (Kategorie 'AWL, FUP, KOP') dient dem Einfügen eines Subnetzwerks innerhalb des bestehenden, und zwar oberhalb der aktuellen Cursor-Position.

12.13.32 Leitungsverzweigung unterhalb einfügen

Symbol:

Dieser Befehl (Kategorie 'AWL, FUP, KOP') dient dem Einfügen eines Subnetzwerks innerhalb des bestehenden, und zwar unterhalb der aktuellen Cursor-Position.

12.13.33 Parameter aktualisieren

Tastaturkürzel: <Strg> + <U>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') kann im FUP/KOP/AWL-Editor verwendet werden, um die Eingangs- und Ausgangsparameter eines bereits eingefügten Bausteins zu aktualisieren, wenn sich dessen Schnittstelle geändert hat (wenn z.B. ein Ausgang hinzugefügt wurde).

Die vorliegenden Anschlüsse an den verbleibenden Ein- oder Ausgängen bleiben erhalten bzw. wenn ein Ein- oder Ausgang hinzukommt, erscheint dieser mit "???" belegt und kann dann zugewiesen werden.

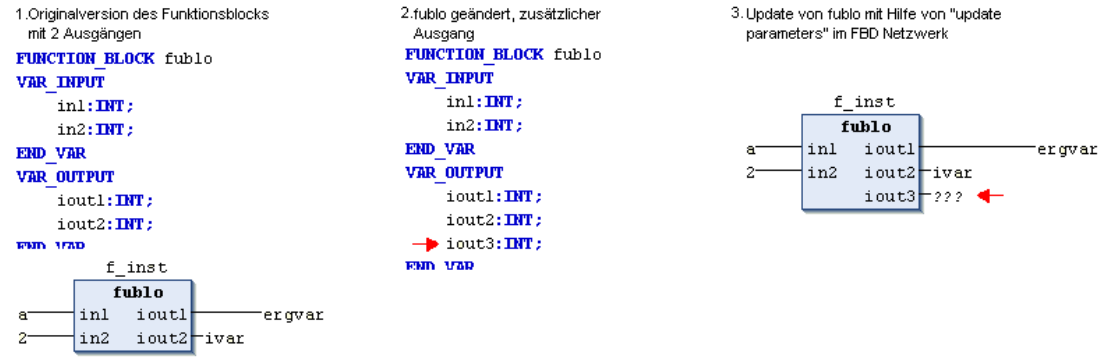
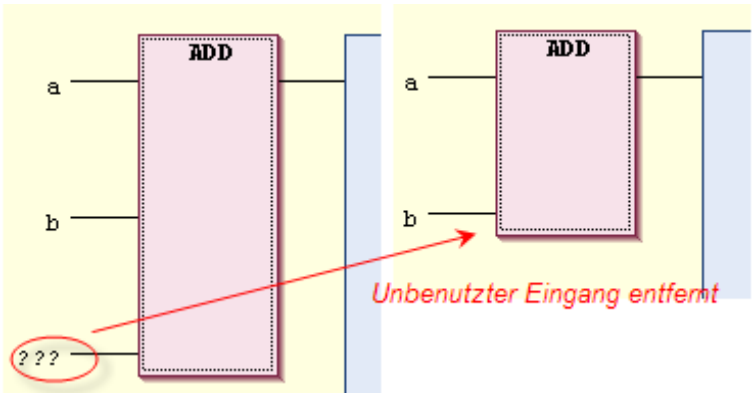


Abb. 118 Beispiel, Parameter aktualisieren

12.13.34 Nicht verwendete FB-Aufruf-Parameter entfernen

Symbol:

Dieses Kommando (Kategorie 'FUP / KOP / AWL') ist nur bei Implementierung in FUP verfügbar. Bei seiner Ausführung werden alle unbenutzten Ein- oder Ausgänge des markierten Bausteins entfernt, also alle Ein- und Ausgänge, die durch '???' gekennzeichnet sind. Die Mindestanzahl notwendiger Ein- oder Ausgänge für einen Bausteinen bleibt dabei jedoch in jedem Fall erhalten.



12.13.35 Ansicht: Als Funktionsbausteinsprache anzeigen

Tastaturkürzel: <Strg> + <1>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') kann im Offline- und Online-Modus eines Bausteins verwendet werden um in die FUP-Ansicht des FUP/KOP/AWL-Editor zu wechseln.

Er bewirkt, dass ein KOP- oder AWL-Netzwerk in ein FUP-Netzwerk konvertiert wird. Beachten Sie allerdings, dass einige spezielle Elemente **nicht konvertiert werden können** und somit weiterhin nur in der ursprünglichen Editor-Ansicht verfügbar bleiben. Außerdem kann es vorkommen, dass Konstrukte nicht eindeutig konvertiert werden können.

Um in die KOP-Ansicht zurück zu schalten, verwenden Sie Ansicht: Als Kontaktplan anzeigen.

Um in die AWL-Ansicht zurück zu schalten, verwenden Sie Ansicht: Als Anweisungsliste anzeigen.

Achtung: Eine fehlerfreie Konvertierung setzt syntaktisch korrekten Code voraus. Anderenfalls können Teile der Implementierung verloren gehen.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.36 Ansicht: Als Kontaktplan anzeigen

Tastaturkürzel: <Strg> + <2>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') kann im Offline- und Online-Modus eines Bausteins verwendet werden um in die KOP-Ansicht des FUP/KOP/AWL-Editor zu wechseln.

Er bewirkt, dass ein AWL- oder FUP-Netzwerk in ein KOP-Netzwerk konvertiert wird. Beachten Sie allerdings, dass einige spezielle Elemente **nicht konvertiert werden können** und somit weiterhin nur in der ursprünglichen Editor-Ansicht verfügbar bleiben. Außerdem kann es vorkommen, dass Konstrukte nicht eindeutig konvertiert werden können.

FUP-Elemente, die nicht als KOP-Elemente dargestellt werden können (z.B. XOR) werden als FUP-Bausteine ins KOP-Netzwerk übernommen.

Um in die FUP-Ansicht zurück zu schalten, verwenden Sie Ansicht: Als Funktionsbausteinsprache anzeigen.

Um in die AWL-Ansicht zurück zu schalten, verwenden Sie Ansicht: Als Anweisungsliste anzeigen.

Achtung: Eine fehlerfreie Konvertierung setzt syntaktisch korrekten Code voraus. Anderenfalls können Teile der Implementierung verloren gehen.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.13.37 Ansicht: Als Anweisungsliste darstellen

Tastaturkürzel: <Strg> + <3>

Dieser Befehl (Kategorie 'AWL, FUP, KOP') kann im Offline- und Online-Modus eines Bausteins verwendet werden um in die AWL-Ansicht des FUP/KOP/AWL-Editor zu wechseln. Er bewirkt, dass ein KOP- oder FUP-Netzwerk in ein AWL-Netzwerk konvertiert wird.

Hinweis:
Beachten Sie für die Konvertierung:
Es gibt Elemente, die nicht konvertiert werden können, deshalb erscheint das betroffene Netzwerk weiterhin in der Original-Editor-Ansicht. Beispiel: Operatoren CALC, CALCN. Im Fall von Syntaxfehlern ist ebenfalls keine Konvertierung möglich; eine entsprechende Fehlermeldung wird ausgegeben.
Einige Konstrukte können nicht eindeutig konvertiert werden und werden deshalb beim Umschalten zwischen AWL und FUP/KOP "normalisiert". Dies betrifft Negationen und explizite/implizite Zuweisungen von Funktionsblock-Ein- und Ausgängen.

Um in die FUP-Ansicht zurück zu schalten, verwenden Sie Ansicht: Als Funktionsbausteinsprache anzeigen.

Um in die KOP-Ansicht zurück zu schalten, verwenden Sie Ansicht: Als Kontaktplan anzeigen.

Achtung: Eine fehlerfreie Konvertierung setzt syntaktisch korrekten Code voraus. Anderenfalls können Teile der Implementierung verloren gehen.

Hinweis: Die Darstellung der Elemente in FUP/KOP/AWL-Netzwerken wird in den 'FUP, KOP und AWL Optionen' konfiguriert.

12.14 Suchen/Ersetzen

Die Befehlskategorie 'Suchen/Ersetzen' enthält Befehle für das Suchen und Ersetzen von Zeichenfolgen im Projekt.

12.14.1 Suchen/Ersetzen Befehle

Das Plug-In 'Find/Replace' stellt Befehle für die Befehlskategorie Suchen/Ersetzen bereit, die verwendet werden können, um im gesamten Projekt nach Zeichenfolgen zu suchen bzw. Zeichenfolgen durch andere zu ersetzen.

Standardmäßig sind die Befehle im Untermenü Suchen & Ersetzen im Bearbeiten-Menü zu finden. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

Die Befehle:

- Suchen
- Ersetzen
- Weitersuchen
- Weitersuchen Auswahl
- Vorheriges suchen
- Vorheriges suchen / Auswahl

12.14.2 Suchen

Symbol:

Standard Tastaturkürzel: <Strg> + <F>

Verwenden Sie diesen Befehl (Kategorie Suchen/Ersetzen) um das Projekt nach einer bestimmten Zeichenfolge zu durchsuchen. Die Suche durchläuft alle editierbaren Stellen in den Objekten des Projekts.

Der Befehl öffnet den 'Suchen'-Dialog. Hier wird eingegeben welche Zeichenfolge (String) gesucht werden soll, wo sie gesucht werden soll und ob die gefundenen Stellen auf einmal oder nacheinander angezeigt werden sollen. Es kann auch jederzeit zum 'Ersetzen'-Dialog umgeschaltet werden.

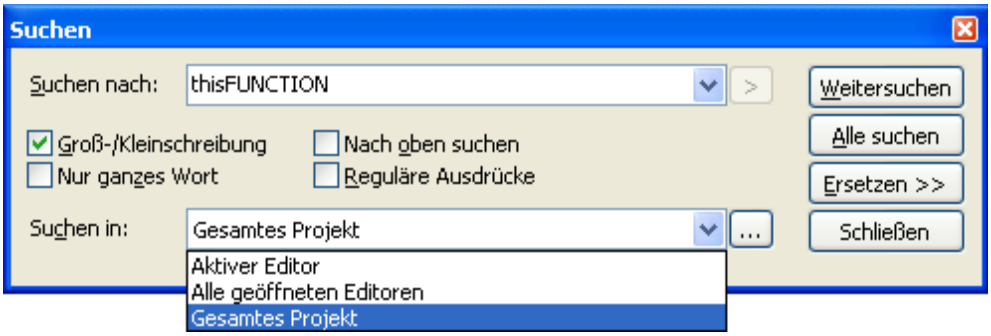


Abb. 119 'Suchen'-Dialog

Suchen nach: Geben Sie hier die Zeichenfolge ein, nach der gesucht werden soll (Such-String). Die Auswahlliste über Schaltfläche bietet alle seit dem Start des Programmiersystems bereits eingegebenen Zeichenfolgen zur Wiederwahl an.

Aktivieren Sie die gewünschten Suchoptionen:

Groß-/Kleinschreibung: Die Suche beachtet Groß-/Kleinschreibung des Such-Strings.

Nur ganzes Wort: Nur Strings, die genau die gesuchte Zeichenfolge wiedergeben, werden gefunden.

Nach oben suchen: Der angegebene Suchraum wird nach oben durchlaufen. Um wieder nach unten zu suchen, deaktivieren Sie die Option.

Reguläre Ausdrücke verwenden: Über die Schaltfläche > erhalten Sie Unterstützung bei der Eingabe von regulären Ausdrücken (Regular Expressions, RegExp, Standard für die Suche von Strings nach Mustern) für die Suche nach bestimmten Zeichenfolgen. Die gebräuchlichsten RegExp's werden unterstützt und sind in Untermenüs sortiert: Special characters (spezielle Zeichentypen), Repetitions (Wiederholungen), Alternatives (Alternativen), Groups (Gruppen) und Others (Andere).

Suchen in: Geben Sie hier an, welche Objekte nach der Zeichenfolge durchsucht werden sollen. Wählen Sie dazu entweder eine der Optionen aus der Liste ( ) oder öffnen Sie über Schaltfläche ... den folgenden Dialog zur Definition des Suchraums:

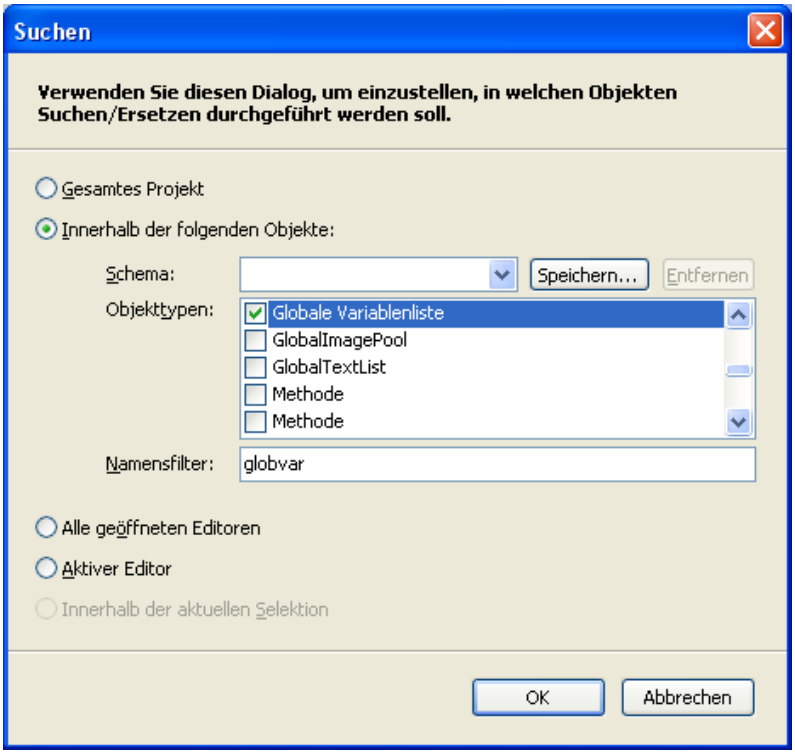


Abb. 120 Dialog zur Angabe des Suchraums

Aktivieren Sie eine der folgenden Optionen:

- **Gesamtes Projekt**: Alle editierbaren Stellen in allen Objekten des Projekts werden durchsucht.
- **Innerhalb der folgenden Objekte**: Nur die editierbaren Stellen innerhalb der hier definierten Objekte werden durchsucht:
  - **Objekttypen**: Setzen Sie einen Haken bei den Objekttypen, die durchsucht werden sollen.
  - **Namensfilter**: Optional kann hier ein Namensfilter für die zu durchsuchenden Objekte gesetzt werden, wobei Platzhalter "*" verwendet werden kann. Beispiel: Wenn alle Objekte durchsucht werden sollen, die in ihrem Namen "CAN" enthalten, geben Sie hier als Filter "*CAN*" ein..

– **Schema**: Sie können die augenblickliche Suchkonfiguration speichern. Drücken Sie dazu die Schaltfläche **Speichern...** und geben im Dialog **Suchschema speichern** einen Namen für die Konfiguration ein. Alle gespeicherten Schemata sind dann in der Auswahlliste ( ) des 'Schema'-Feldes verfügbar. Das gerade ausgewählte Schema kann über Schaltfläche **Entfernen** auch wieder gelöscht werden.

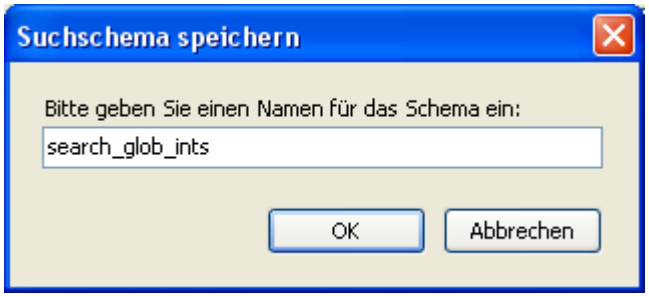


Abb. 121 Dialog 'Suchschema speichern'

- **Alle geöffneten Editoren**: Alle Editoren, die gerade in einem Fenster geöffnet sind, werden durchsucht.
- **Aktiver Editor**: Nur der Editor, in dem gerade der Cursor steht, wird durchsucht.
- **Innerhalb der aktuellen Selektion**: Nur der gerade in einem Objekt ausgewählte Text wird nach dem Such-String durchsucht.

Wenn alle Suchoptionen definiert sind, drücken Sie Schaltfläche...

... Weitersuchen, um jeweils zur nächsten gefundenen Stelle zu springen. Die jeweils betroffenen Editoren werden geöffnet und der gefundene String wird hervorgehoben angezeigt.

... **Alle suchen**, um eine Liste aller gefundenen Stellen im Meldungsfenster zu erhalten. Innerhalb der Statuszeile wird hierbei der Fortschritt des Suchprozesses angezeigt und kann ggf. mit Hilfe der in der Statuszeile angezeigten Taste **Abbrechen** vorzeitig beendet werden. Nach Abschluss der Suche werden die folgenden Informationen werden zu jeder Stelle angegeben:

Beschreibung: Ausdruck, der den Such-String enthält

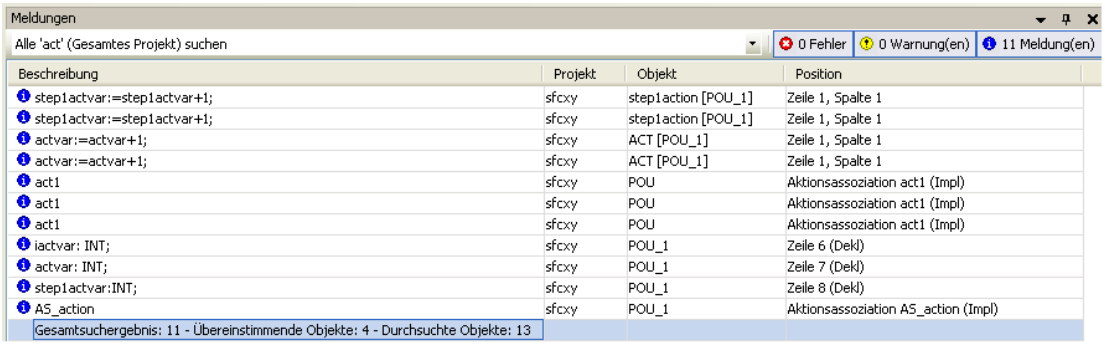
Projekt: Projektname

Objekt: Objektname

Position: Position (z.B. Zeilennummer) innerhalb des Objekts; in Klammern steht "Decl" für den Deklarationsteil bzw. "Impl" für den Implementationsteil des Editors.

Unterhalb der Liste wird folgendes angegeben: Gesamtzahl der gefundenen Stellen, Zahl der Objekte, in denen der String gefunden wurde und Gesamtzahl der durchsuchten Objekte.

Beispiel: Suchergebnisse für String "b":



Wenn Sie den gefundenen String durch einen anderen ersetzen wollen, drücken Sie Schaltfläche **Ersetzen**, um zum Ersetzen-Dialog zu gelangen.

12.14.3 Ersetzen

Symbol:

Standard Tastaturkürzel: <Strg> + <H>

Dieser Befehl (Kategorie Suchen/Ersetzen) öffnet den 'Ersetzen'-Dialog, bei dem es sich um einen erweiterten Suchen-Dialog handelt.

Setzen Sie wie im Suchen-Dialog die Optionen zum Finden der Zeichenfolge, die sie durch eine andere ersetzen wollen. Geben Sie zusätzlich bei **Ersetzen mit:** die neue Zeichenfolge ein und verwenden dann eine der 'Ersetzen'-Schaltflächen:

Ersetzen: Damit wird jeweils der nächste gefundene String im Editor hervorgehoben und ersetzt (Schrittweises Ersetzen).

Alles ersetzen: Alle gefundenen Strings werden auf einmal ersetzt, ohne dass sie in den Editoren angezeigt werden.

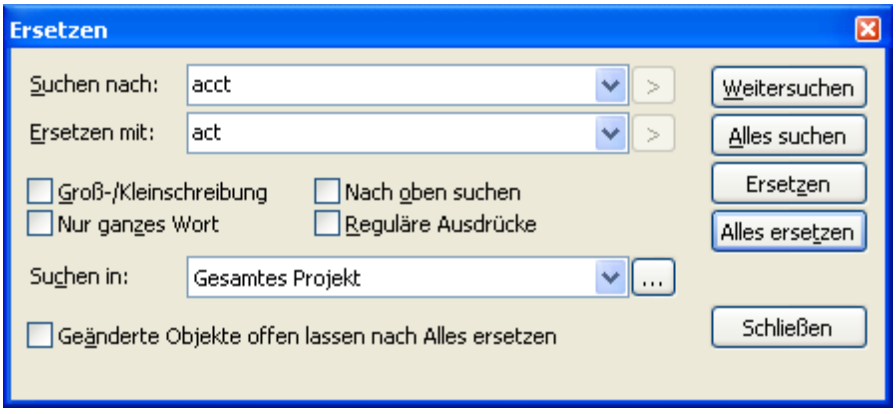


Abb. 122 'Ersetzen'-Dialog

12.14.4 Weitersuchen

Symbol:

Standard Tastaturkürzel: <F3>

Dieser Befehl (Kategorie Suchen/Ersetzen) steht bereit, um zur nächsten gefundenen Position zu gelangen, nachdem mit den Befehlen Suchen oder Ersetzen nach einer bestimmten Zeichenfolge im Projekt gesucht wurde.

12.14.5 Weitersuchen / Auswahl

Standard Tastaturkürzel: <Strg>+<F3>

Dieser Befehl (Kategorie Suchen/Ersetzen) steht bereit, um zur nächsten gefundenen Position einer Zeichenfolge im Projekt zu gelangen, die der gerade im Editor markierten entspricht.

12.14.6 Vorheriges suchen

Symbol:

Standard Tastaturkürzel: <Umsch>+<F3>

Dieser Befehl (Kategorie Suchen/Ersetzen) steht bereit, um zur vorherigen gefundenen Position zu gelangen, nachdem mit den Befehlen Suchen oder Ersetzen nach einer bestimmten Zeichenfolge im Projekt gesucht wurde.

12.14.7 Vorheriges suchen / Auswahl

Standard Tastaturkürzel: <Umsch>+<Strg>+<F3>

Dieser Befehl (Kategorie Suchen/Ersetzen) steht bereit, um zur vorherigen gefundenen Position einer Zeichenfolge im Projekt zu gelangen, die der gerade im Editor markierten entspricht.

12.15 Datei

Die Befehlskategorie 'File' enthält Befehle für das Handhaben der Projektdatei.

12.15.1 Kategorie "Datei"

Befehle der Kategorie Datei

Das Plug-In 'File Commands' stellt Befehle für Befehlskategorie 'Datei' bereit, die für die Handhabung der Projektdatei verwendet werden können.

Es hängt vom aktuell eingestellten Feature-Set ab, ob alle Befehle dieser Kategorie verfügbar sind.

Standardmäßig sind die meisten dieser Befehle im Datei-Menü enthalten, manche im Projekt-Menü.. Falls erforderlich, kann die Menüstruktur mit Hilfe des 'Anpassen'-Dialogs umkonfiguriert werden.

Die Befehle:

- Neues Projekt...
- Projekt öffnen...
- Projekt schließen
- Projekt speichern
- Projekt speichern unter...
- Projekt speichern und ins Bibliotheks-Repository installieren
- Projekt als übersetzte Bibliothek speichern
- Projektarchiv: Archiv speichern/versenden...
- Projektarchiv: Archiv extrahieren...
- Projekt kopieren
- Zuletzt verwendete Projekte...
- Beenden
- Vergleichen...
- Projektumgebung
- Projektinformationen
- Projekteinstellungen

Neues Projekt...

Symbol:

Default Shortcut: <Strg>+<N>

Dieser Befehl (Kategorie 'Datei') öffnet den Dialog 'Neues Projekt' zum Anlegen einer neuen Projektdatei.

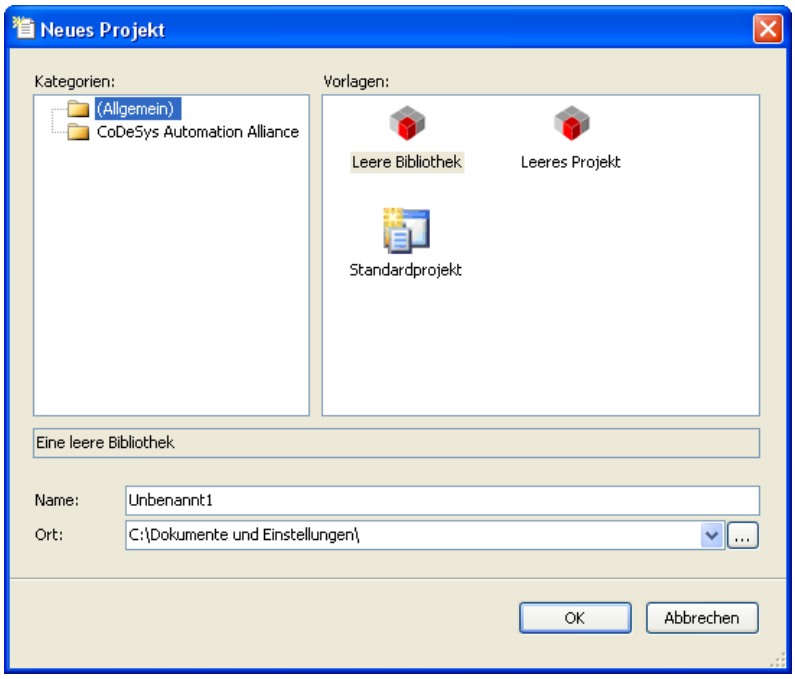


Abb. 123 Dialog - Neues Projekt

Kategorien: Hier werden die in Kategorien zusammengefaßten Vorlagen und Projekt-Wizards in einer Baumstruktur angeboten. Für die Kategorie, die ausgewählt ist, werden die jeweiligen Vorlagen und Wizards rechts im Vorlagen-Fenster angezeigt.

Vorlagen: Die hier auswählbaren Vorlagen/Wizards gehören zu der gerade im Kategorien-Fenster ausgewählten Kategorie. Eine "Vorlage" bestimmt die Basiskonfiguration einer Projektdatei.
Standardmäßig ist hier die Vorlage "Standardprojekt", für ein "Leeres Projekt" *.project und für eine "Leere Bibliothek" *.library enthalten. Das Standardprojekt wird von einem Wizard unterstützt. Die leeren Vorlagen enthalten keine bereits eingefügten Objekte oder Geräte.

Name: Hier wird der Name des neuen Projekts definiert. Der Default-Name wird durch die verwendete Vorlage bestimmt (üblicherweise "Unbekannt<n>") und enthält eine fortlaufende Nummer, um die Eindeutigkeit des Projektnamens im Dateisystem zu gewährleisten. Der vorgegebene Name kann unter Berücksichtigung der Dateipfad-Konventionen des lokalen Betriebssystems geändert werden. Eine Dateierweiterung (z.B. ".project") kann hinzugefügt werden, per Default wird automatisch durch die gewählte Vorlage die passende Erweiterung angehängt.

Ort: Speicherort für die neue Projektdatei. Der Standardpfad wird durch die gewählte Vorlage bestimmt. Der Windows Standard-Browser kann zum Eingeben eines gewünschten Pfads über die Schaltfläche ... geöffnet werden. Über Schaltfläche kann auch die Historie bereits eingegebener Pfade zu Hilfe genommen werden.

OK: Ein neues Projekt wird entsprechend der vorgenommenen Einstellungen angelegt. Wenn Angaben fehlen, wird dies am entsprechenden Eingabefeld mit einem Fehlersymbol angezeigt. Wenn der Cursor auf ein Fehlersymbol gesetzt wird, gibt ein Tooltip Hinweise, was zu tun ist. Wenn momentan bereits ein anderes Projekt geöffnet ist, werden Sie nun gefragt, ob dieses gespeichert und geschlossen werden soll, bevor das neue geöffnet wird.

In jedem Fall wird der Name des neuen Projekts jetzt in der Titelleiste des »PLC Designer«-Rahmenfensters angezeigt. Beachten Sie: Wenn ein Sternchen (*) hinter dem Namen erscheint, heißt das, dass das Projekt seit dem letzten Speichern verändert wurde.

Wizard für das Standardprojekt:

Wenn Sie die Vorlage Standardprojekt auswählen und mit **OK** bestätigen, so erscheint ein Wizard Dialog:

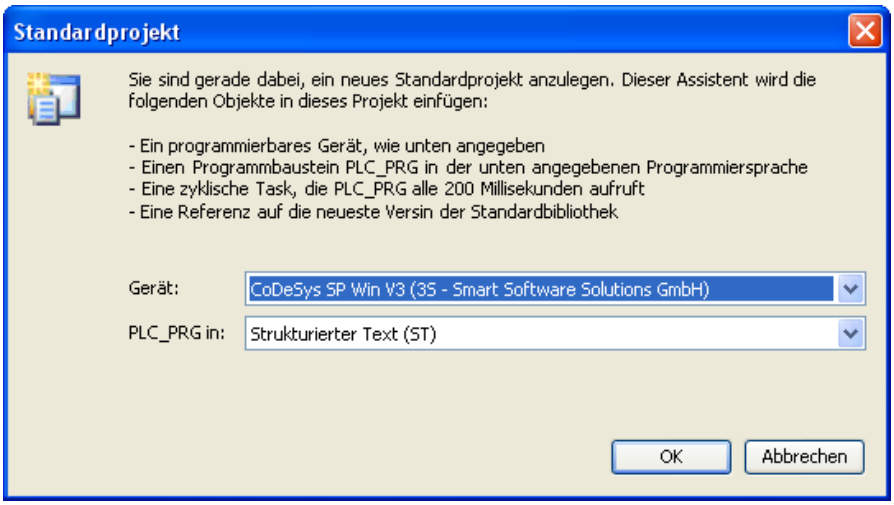


Abb. 124 Dialog - Standardprojekt Wizard

Eine Bestätigung mit **OK** wird folgendes Standardprojekt ergeben:

Erzeugt wird ein Projekt mit der aus der Liste gewählten **Steuerung**, einer **Applikation** "Application", einem **Bibliotheksverwalter**, einem Programm POU "**PLC_PRG**" und einer **Taskkonfiguration** mit der Task "**MainTask**" im Gerätebaum. Der Editor von PLC_PRG entspricht der Programmiersprache, die aus der Liste bei **PLC_PRG in** ausgewählt wurde. Die Task ist zyklisch und ruft PLC_PRG alle 200 Millisekunden auf. Im Bibliotheksverwalter ist die aktuell installierte Version der Bibliothek **Standard.library** enthalten.

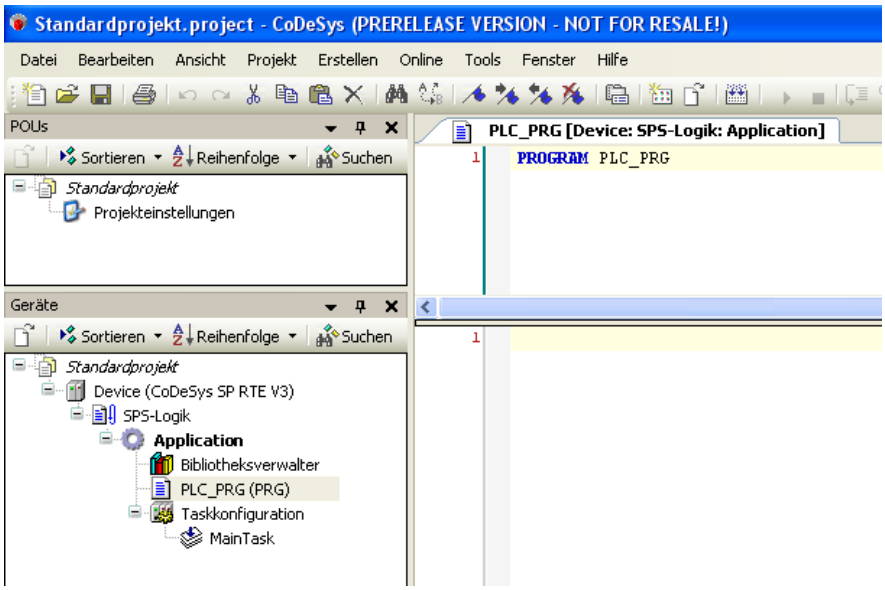


Abb. 125 Beispiel - Standard Project

Projekt öffnen...

Symbol:

Default Shortcut <Strg>+<O>

Batch command: file open projectpath [ConverterGUID] (not yet implemented)

Dieser Befehl (Kategorie 'Datei') öffnet den Standard-Browserdialog, über den eine Projektdatei gesucht werden kann, die im Programmiersystem geöffnet werden soll. Wenn ein entsprechender Konverter installiert ist, können auch Projekte anderen Formats geöffnet werden.

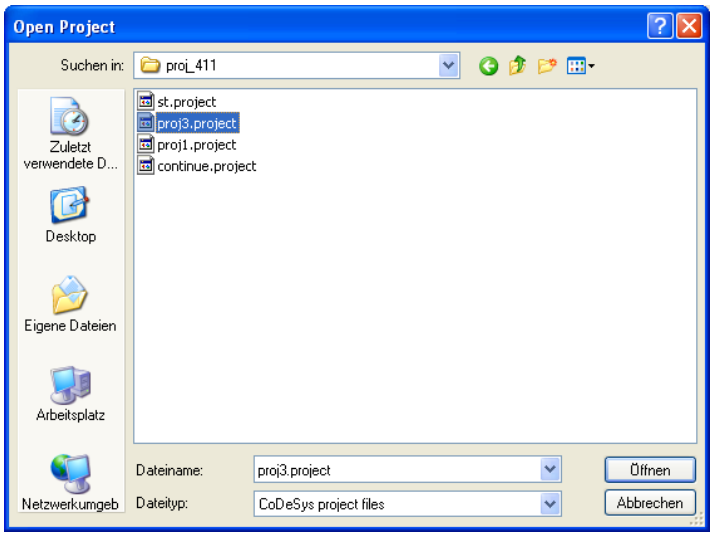


Abb. 126 Dialog 'Projekt öffnen'

Dateityp: Standardmäßig können Sie den Filter auf einen der folgenden Dateitypen setzen:

- »PLC Designer« Projekte: »PLC Designer« Projekte ab V3.0, Erweiterung ".project"
- »PLC Designer« Bibliotheken: »PLC Designer« Bibliotheken ab V3.0, Erweiterung ".library"
- Alle Dateien: alle vorhandenen Dateien

zusätzlich werden, wenn der entsprechende Projektkonverter installiert ist, folgende Dateitypen angeboten:

- »PLC Designer« Projekte (vor V3.0), Erweiterung ".pro"
- »PLC Designer« Bibliotheken (vor V3.0), Erweiterung ".lib"

Öffnen: Die gewählte Projektdatei wird geöffnet bzw. konvertiert und dann geöffnet. Folgende Szenarien sind möglich:

(1) Ein anderes Projekt ist noch geöffnet

(2) Projekt wurde mit anderer V3.x-Version gespeichert

(3) Projekt wurde nicht mit V3.x gespeichert

(4) Projekt wurde nicht regulär beendet und Automatisch Speichern war aktiviert

(5) Projekt ist schreibgeschützt

(6) Es handelt sich um eine Bibliothek, die in einem Bibliotheks-Repository installiert ist und aus diesem aufgerufen wird

1) Es ist noch ein anderes Projekt geöffnet: Sie werden gefragt, ob dies gespeichert und geschlossen werden soll.

2) Das Projekt wurde mit älteren Version von »PLC Designer« V3 gespeichert:
Wenn sich das **Speicherformat** unterscheidet, weil das geöffnete Projekt mit einer älteren »PLC Designer« V3 Version gespeichert worden war, sind zwei Fälle möglich:

1. Wenn das Projekt nicht im Speicherformat des aktuell verwendeten Programmiersystems speicherbar ist, muss eine Aktualisierung erfolgen, um weiter am Projekt arbeiten zu können. Lassen Sie sich nicht durch den Ausdruck "Die durchgeführten Änderungen..." verwirren, an dieser Stelle sind damit interne Aktionen verschiedener Komponenten während des Ladens des Projekts gemeint.

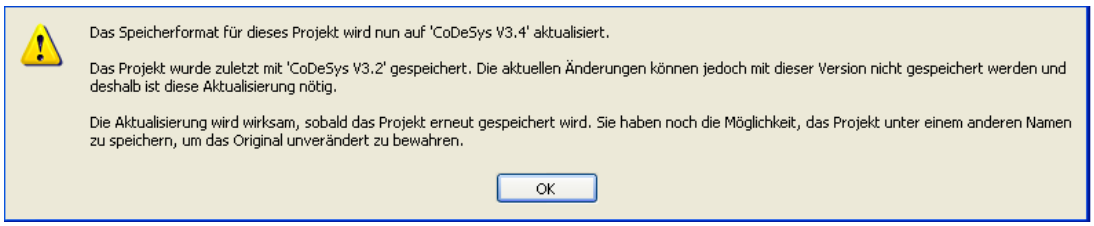


Abb. 127 Meldungsbox mit Hinweis auf erforderliche Speicherformat-Aktualisierung, Beispiel

2. Wenn das Projekt im bisherigen Speicherformat weiterhin speicherbar ist, kann jetzt entschieden werden, ob dieses Format beibehalten werden soll - allerdings muss dann mit möglichen Datenverlusten gerechnet werden - oder ob das Speicherformat aktualisiert werden soll, woraufhin allerdings das Projekt dann nicht mehr mit der alten Programmiersystemversion geöffnet werden kann.

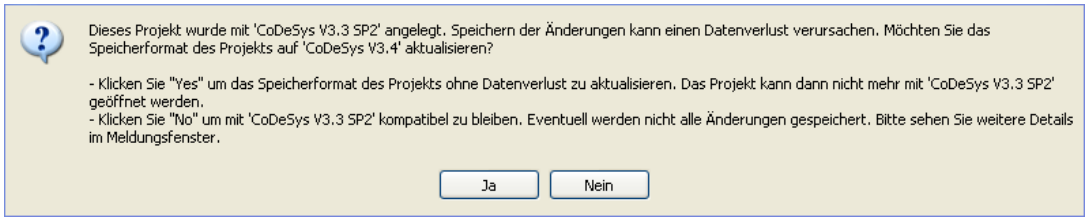


Abb. 128 Meldungsbox mit Angebot einer Speicherformat-Aktualisierung, Beispiel

Projektumgebung

Neben dem Speicherformat können sich auch die **Versionen der Bibliotheken** (der explizit eingefügten), des **Visualisierungsprofils** und der **Compiler-Version** des zu öffnenden Projekts von den mit dem aktuellen Programmiersystem installierten unterscheiden. In diesem Fall öffnet der Dialog **Projektumgebung** mit den entsprechenden Unterdialog(en), wo Versionsaktualisierungen angeordnet werden können, wenn neuere Versionen auf dem System installiert sind. Zu beachten: Wenn an dieser Stelle noch keine Aktualisierung vorgenommen wird, kann dies jederzeit später noch gemacht werden. Die Projektumgebung-Dialoge sind immer über den Befehl 'Projektumgebung' zugänglich. Siehe eine Beschreibung dazu unter Projektumgebung.

Hinweis: Wenn ein Projekt geöffnet wird, das mit einer älteren Version des Programmiersystems erstellt wurde, und für das die Compiler-Version "Neueste.." ist (Projekteinstellungen), während hier im neuen Programmiersystem die Projektumgebungseinstellung für die Compile-Version "Nicht aktualisieren" ist, dann wird die zuletzt im alten Projekt verwendete Compile-Version weiter verwendet (nicht die "Aktuelle" in der neuen Umgebung).

3) Das Projekt wurde nicht mit Programmiersystem »PLC Designer« V3.x gespeichert:

- Wenn beim Auswählen des zu öffnenden Projekts der Dateifilter gezielt gesetzt wurde, wird automatisch der entsprechende Konverter - wenn verfügbar - verwendet und das Projekt in das aktuelle Format gebracht. Die Konvertierung läuft konverter-spezifisch ab. In der Regel wird man gegebenenfalls aufgefordert, die Behandlung von eingebundenen Bibliotheken oder Geräte-Referenzen zu definieren - wie unten beschrieben: Konvertierung einer Bibliotheksreferenz, Gerätekonvertierung.

Hinweis: Beachten Sie für den »PLC Designer« V2.3 Konverter: Die Anpassung eines V2-Projektes an die V3 Syntax kann beim Import nur gelingen, wenn der Konverter das Projekt fehlerfrei übersetzen kann.

- Wenn beim Auswählen des zu öffnenden Projekts der Dateityp 'Alle Dateien' eingestellt war, ist kein Konverter aktiviert und es wird der Dialog '**Projekt konvertieren**' geöffnet, in dem die Konvertierung des Projekts durch Auswählen einer der Optionen explizit angestoßen werden muss.

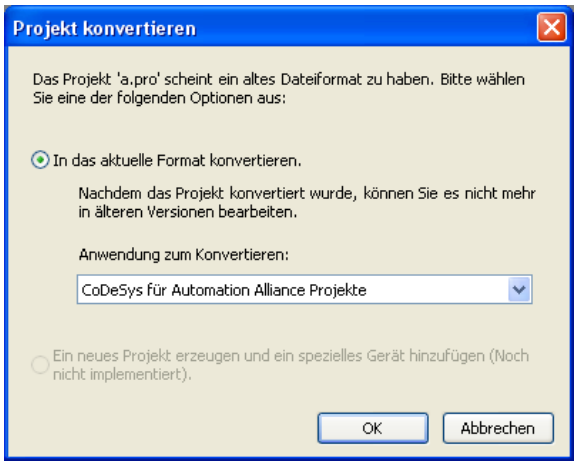


Abb. 129 Dialog 'Projekt konvertieren'

- **In das aktuelle Format konvertieren**: Wählen Sie aus der Auswahlliste den Konverter, der verwendet werden soll (**Anwendung zum Konvertieren**). Beachten Sie, dass nach dem Konvertieren das Projekt nicht mehr in der alten Version geöffnet werden kann.
- **Ein neues Projekt erzeugen und ein spezielles Gerät hinzufügen** (Noch nicht implementiert)

Hinweis: Beachten Sie für den »PLC Designer« V2.3 Konverter: Der in den V2.3-Projektoptionen eingestellte Projektverzeichnispfad sowie die Projektinformationen werden in den Dialog 'Projektinformationen' übernommen.

- Falls im Projekt Bibliotheken eingebunden sind, für die noch kein "Konvertierungs-Mapping" in den Bibliotheks-Optionen gespeichert ist, erscheint der Dialog **Konvertierung einer Bibliotheksreferenz**, in dem definiert wird, wie diese Referenz konvertiert werden sollen:

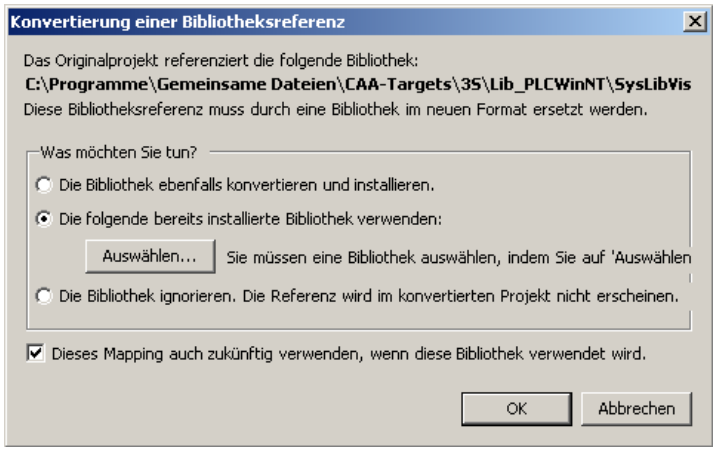


Abb. 130 Dialog 'Konvertierung einer Bibliotheksreferenz'

(Generelle Informationen zur Bibliotheksverwaltung finden Sie unter Bibliotheksverwaltung):

Die Bibliothek ebenfalls konvertieren und installieren.: Wenn diese Option aktiviert ist, wird die eingebundene Bibliothek in das neue Format übergeführt und bleibt im Projekt referenziert. Sie wird automatisch im Bibliotheks-Repository in Kategorie 'Sonstige" installiert und weiterhin verwendet. Wenn die Bibliothek nicht die für eine Installation nötigen Projektinformationen (Titel, Version) mitbringt, werden Sie nun aufgefordert, diese im Dialog 'Projektinformationen eingeben' nachzutragen.

Die folgende bereits installierte Bibliothek verwenden: Aktivieren Sie diese Option, wenn die eingebundene Bibliothek durch eine andere ersetzt werden soll, die bereits auf dem lokalen System installiert ist. Die Schaltfläche **Auswählen** öffnet dazu den Dialog **Bibliothek auswählen**. Hier können Sie die gewünschten Version einer der installierten Bibliothek auswählen. Dies entspricht der Konfiguration des Versions-Handlings im Bibliothekseigenschaften-Dialog. "*" bedeutet, dass immer die neueste Version der Bibliothek, die auf dem System verfügbar ist, im Projekt verwendet wird. Die Liste der verfügbaren Bibliotheken ist genauso strukturiert, wie des für den Bibliotheks-Repository Dialog beschrieben ist. Es kann nach Firma und Kategorie sortiert werden.

Die Bibliothek ignorieren. Die Referenz wird im konvertierten Projekt nicht erscheinen.: Wenn diese Option aktiviert ist, wird die Bibliotheksreferenz entfernt und die Bibliothek ist im konvertierten Projekt nicht mehr eingebunden.

Dieses Mapping auch zukünftig verwenden, wenn diese Bibliothek auftritt.: Wenn diese Option aktiviert ist, werden die hier im Dialog vorgenommen Einstellungen auch für künftige Projekt-Konvertierungen angewendet, sobald die betreffende Bibliothek referenziert ist.

Im konvertierten Projekt sind die Bibliotheksreferenzen im globalen Bibliotheksverwalter im POUs-Fenster definiert.

Nach Abschluß der Konvertierung der Bibliotheks-Referenzen wird die Projekt-Konvertierung wie oben.beschrieben mit Dialog 'Projekt öffnen' weitergeführt.

- Wenn ein V2.3 Projekt geöffnet wird, das ein Gerät (Zielsystem) referenziert, für das noch kein "Konvertierungs-Mapping" in den »PLC Designer« 2.3 Konverter Optionen definiert ist, wird der Dialog '**Gerätekonvertierung**' geöffnet, in dem man festlegen kann, ob und wie die alten Geräte-Referenzen durch aktuelle ersetzt werden sollen:

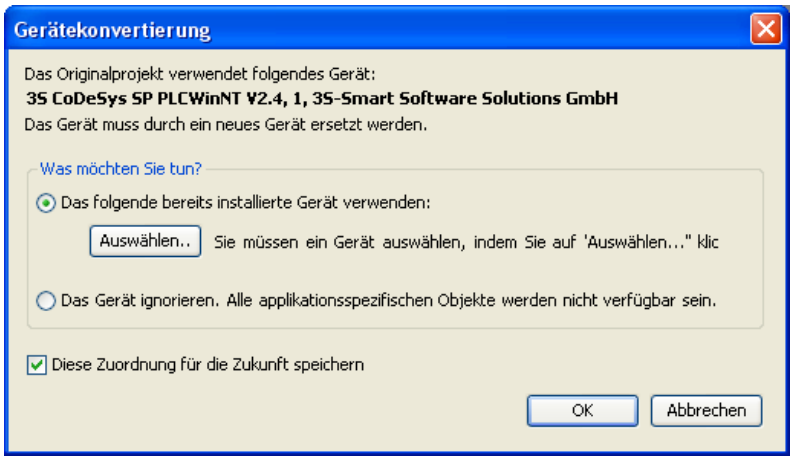


Abb. 131 Dialog 'Gerätekonvertierung'

(Für allgemeine Informationen zu Geräten sehen Sie bitte unter Geräte, Gerätebaum)

Das ursprünglich verwendete Gerät wird angezeigt und Sie müssen entscheiden, was mit der Geräte-Referenz im neuen Projekt geschehen soll. Wählen Sie eine der folgenden Optionen:

Das folgende bereits installierte Gerät verwenden: Die Schaltfläche **Auswählen** öffnet den Dialog **Zielsystem auswählen**, in dem eines der aktuell auf dem System installierten Geräte ausgewählt werden kann. Dieses Gerät wird dann anstelle des alten im Gerätebaum des konvertierten Projekts eingefügt.

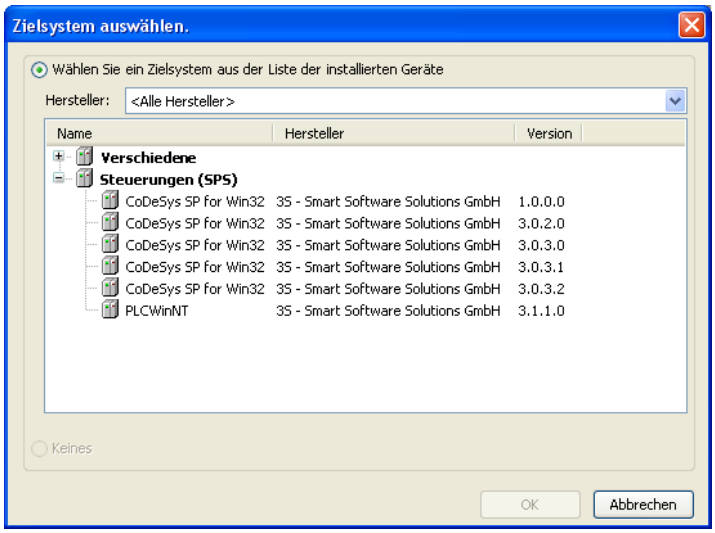


Abb. 132 Dialog 'Zielsystem auswählen'

Aktivieren Sie Option **Wählen Sie ein Zielsystem aus...** um eines der aufgelisteten Geräte auswählen zu können. Die Handhabung des Dialogs (Sortierung nach Hersteller und Kategorien) entspricht dem des Device-Repository Dialogs.

Das Gerät ignorieren. Alle applikationsspezifischen Objekte werden nicht verfügbar sein.: In diesem Fall wird im Gerätebaum des neuen Projekts kein Eintrag für das Gerät angelegt, d.h. dieses wird bei der Konvertierung ignoriert und auch applikationsspezifische Objekte wie z.B. die Taskkonfiguration werden nicht übernommen.

Diese Zuordnung für die Zukunft speichern.: Wenn diese Option aktiviert ist, werden alle Einstellungen des Dialogs, d.h. das dargestellte Konvertierungs-Mapping für das Gerät, in den »PLC Designer« 2.3 Konverter Optionen gespeichert und für künftige Konvertierungen angewendet.

4) Wenn die '**Automatisch Speichern**'-Funktion in den Optionen für 'Laden und Speichern' aktiviert war und »PLC Designer« vor dem nach der letzten Änderung des Projekts ohne Speichern nicht regulär beendet worden war, erhalten Sie nun den Dialog 'Auto Save Backup' zur Handhabung der Sicherungskopie.

5) Ist das Projekt, das geöffnet werden soll, schreibgeschützt, so werden Sie gefragt, ob Sie das Projekt in schreibgeschütztem Modus öffnen wollen oder ob Sie es beschreibbar machen wollen.

6) Wenn versucht wird, ein **Bibliotheksprojekt** zu öffnen, das in einem Bibliotheks-Repository installiert ist, wird eine Fehlermeldung ausgegeben. Das Projekt kann über diesen Pfad nicht geöffnet werden.

Nach Schließen des Dialogs mit **OK** erscheint der Projektname in der Titelzeile des Benutzeroberfläche. Beachten Sie: Wenn ein Sternchen (*) hinter dem Namen erscheint, heißt das, dass das Projekt seit dem letzten Speichern verändert wurde.

Projekt schließen

Batch command: file close (not yet implemented)

Dieser Befehl (Kategorie 'Datei') schließt das Projekt, wobei das Programmiersystem geöffnet bleibt. Wenn das Projekt nicht-gespeicherte Änderungen enthält, werden Sie gefragt, ob diese gespeichert werden sollen.

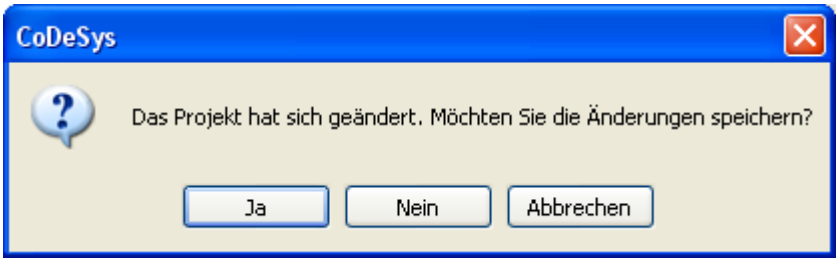


Abb. 133 Dialog mit Abfrage, ob Projekt gespeichert werden soll

Um auch das Programmiersystem zu schließen, verwenden Sie den Befehl Beenden.

Projekt speichern

Symbol:

Default Shortcut <Strg>+<S>

Batch command: file save (not yet implemented)

Dieser Befehl (Kategorie 'Datei') speichert die Projektdatei. Er ist nur verfügbar, wenn das Projekt seit dem letzten Speichern verändert wurde. Dies ist an einem Sternchen (*) hinter dem Projektnamen in der Titelleiste des Programmiersystems erkennbar.

Abb. 134 Titelleiste, verändertes Projekt

Falls das Projekt schreibgeschützt ist, wird der folgende Meldungsdialog geöffnet:

Abb. 135 Meldungsdialog, Projekt ist schreibgeschützt

Speichern als.. ermöglicht einen neuen Projektpfad zu definieren.

Überschreiben entfernt den Schreibschutz von der Projektdatei und speichert.

Sicherungskopie:

Optional kann eine Sicherungskopie der Projektdatei angelegt werden. Wenn die Option **Sicherungskopie erzeugen** im Optionen-Dialog 'Laden und Speichern' aktiviert ist, wird das Projekt bei jedem Speichern in *<projektname>.project* zusätzlich auch in eine Datei *<projektname.backup>* kopiert.

Projekt speichern unter...

Mit diesem Befehl (Kategorie 'Datei') wird das Projekt gespeichert, wobei aber Speicherpfad und Dateityp explizit angegeben werden können. Der Windows Standard-Dialog zum Speichern einer Datei wird zu diesem Zweck geöffnet.

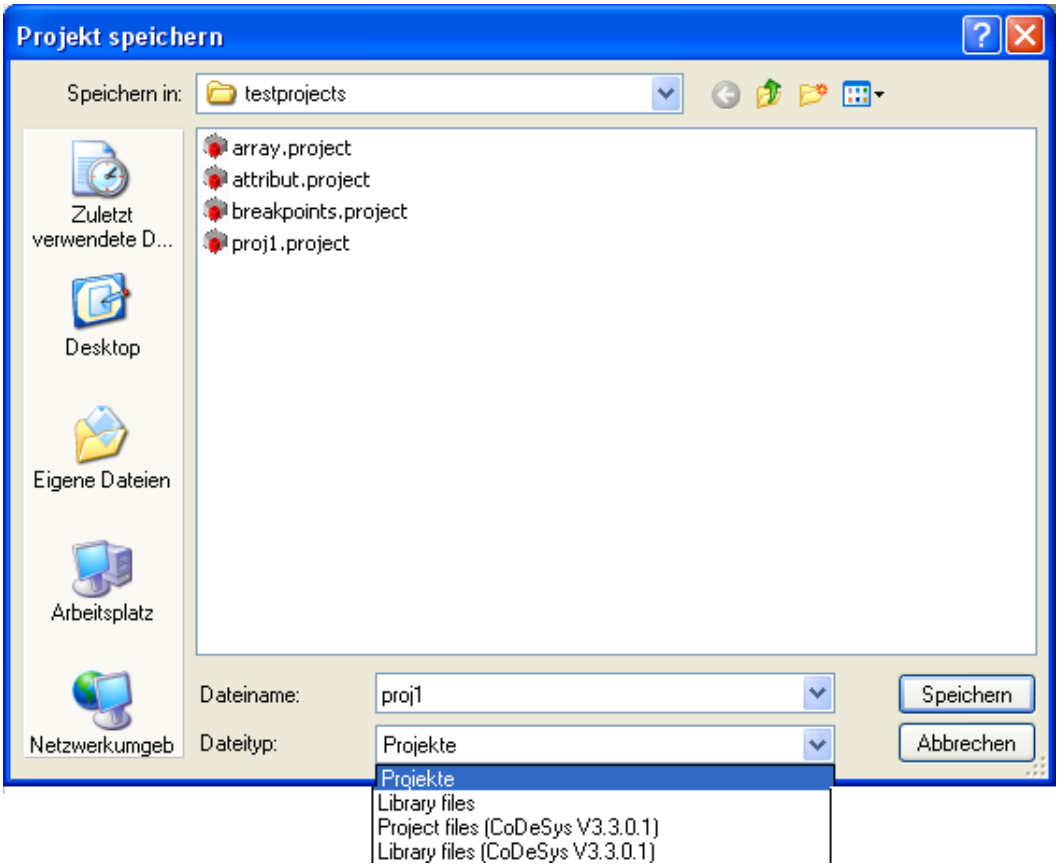


Abb. 136 Dialog 'Projekt Speichern'

1. In den Feldern **Speichern in** und **Dateiname** sind zunächst automatisch der bisherige Speicherort und Dateiname eingetragen und können bei Bedarf geändert werden.

2. Im Feld **Dateityp** wählen Sie einen der verfügbaren Dateitypen aus. Die Auswahlliste enthält für normale Projekte wie auch für Bibliotheksprojekte jeweils die Versionen des Programmiersystems, für die das Projekt gespeichert werden kann. Wenn das Projekt später in einer älteren Version geöffnet werden soll, ist es sinnvoll genau für diese Version zu speichern, da dann sofort über eventuelle Datenverluste informiert wird. Beachten Sie dazu die Standard-Projekteinstellung für das Speicherformat:

   Projektdateien bzw. **Projektdateien (**»PLC Designer« **V<version>)**: Das Projekt wird als »PLC Designer« Projektdatei "<projektname>.project " für die aktuell verwendete bzw. die ausgewählte Version des Programmiersystems gespeichert. .

   Bibliotheksdateien bzw. **Bibliotheksdateien (**»PLC Designer« **V<version>)**: Das Projekt wird als »PLC Designer« Bibliotheksdatei "<projektname>.library" für die aktuell verwendete bzw. die ausgewählte Version des Programmiersystems gespeichert.
   Beachten Sie folgendes bevor Sie ein Projekt als Bibliothek ablegen:

- Wenn Sie die Bibliothek später für die Verwendung in anderen Projekten installiert werden können soll, fügen Sie die erforderlichen Angaben zu Version und Firma in den Projektinformationen ein.
- Beim Speichern des Projekts wird nicht automatisch eine Überprüfung auf Fehler durchgeführt!
- Beachten Sie, dass es **im Gegensatz zu »PLC Designer« V2.3** keine Unterscheidung zwischen "externen" und "internen" Bibliotheken gibt. Aber es kann in den Eigenschaften jedes einzelnen Projektobjekts definiert werden, ob es als "externes" behandelt werden soll.
- Überlegen Sie, ob die erzeugte Bibliothek sofort ins System-Bibliotheks-Repository installiert werden soll. In diesem Fall verwenden Sie Befehl Projekt speichern und ins Bibliotheks-Repository installieren.
- Wenn das Bibliotheksprojekt gegen spätere Änderungen geschützt werden soll, aktivere man OptionFreigegeben im Projektinformationen-Dialog. Dies wird beim ersten späteren Versuch, das Bibliotheksprojekt zu ändern, eine Bestätigung dass dies wirklich gewünscht ist erzwingen.

Wenn bereits eine Datei mit dem definierten Pfad und Namen vorliegt, wird dein **Meldungsdialog** geöffnet, in dem Sie gefragt werden, ob diese Datei mit der neuen ersetzt werden soll. Wenn Sie "Nein" wählen, gelangen Sie zurück zum 'Projekt speichern'-Dialog, um einen anderen Dateinamen oder Speicherort zu definieren.
Wenn Sie das Projekt in einer anderen als der aktuell verwendeten Version des Programmiersystems speichern wollen, werden Sie auf möglichen Datenverlust hingewiesen.

Projekt speichern und ins Bibliotheks-Repository installieren

Symbol:

Mit diesem Befehl (Kategorie 'Datei') wird das Projekt als Bibliothek im "System"-Bibliotheks-Repository gespeichert. Dies ist eine Erweiterung zum Speichern eines Projekts als Bibliotheksdatei über den Befehl Projekt speichern unter..., da die Bibliothek gleichzeitig auf dem lokalen System installiert wird und somit unmittelbar zum Einfügen in einem Projekt - via Bibliotheksmanager- bereitsteht. **Bitte sehen Sie** die Beschreibung zu Befehl Projekt speichern unter... bzw. die allgemeinen Informationen zur Bibliotheksverwaltung um zu wissen, was für die Erstellung einer Bibliothek zu beachten ist.

Projekt als übersetzte Bibliothek speichern

Mit diesem Befehl (Kategorie 'Datei') kann ein Bibliotheksprojekt in verschlüsseltem Format gespeichert werden. Genauer gesagt wird ein verschlüsseltes Abbild des Precompile-Kontexts der Bibliothek abgelegt, was bedeutet, dass die Implementierungen der Bibliotheksbausteine nicht mehr zugänglich und sichtbar sind.

Der Befehl öffnet den Standarddialog zum Speichern einer Datei im Dateisystem, wobei der Dateityp "**Übersetzte »PLC Designer« Bibliotheken**" bereits voreingestellt ist. Automatisch wird der bisherige Dateiname angeboten, kann jedoch verändert werden. Die Dateierweiterung ist "**.compiled-library**".

In der weiteren Handhabung verhalten sich *.compiled-library-Dateien genauso wie *.library-Dateien, können also auf die gleiche Art und Weise installiert und referenziert werden.

So kann zum einen der Quell-Code einer Bibliothek geschützt werden und zum anderen ergibt sich der Vorteil kürzerer Ladezeiten und kleinerer Bibliotheksdateien.

Achtung: Die üblichen Debug-Funktionalitäten können auf eine solche Bibliothek nicht mehr angewendet werden; d.h. "Steppen" in einen Bibliotheksbaustein ist nicht möglich.

Projektarchiv: Archiv speichern/versenden...

Dieser Menüpunkt (Kategorie 'Datei') dient zur Konfiguration und zum Erstellen eines Projektarchivs, in welchem alle im aktuell geöffneten Projekt enthaltenen und referenzierten Dateien zusammengefasst sind. Die Archivdatei (Endung standardmäßig .projectarchive) kann daraufhin entweder gespeichert oder aber als Anhang einer Email verschickt werden. Letzeres ist hilfreich um einem Mitarbeiter alle projektrelevanten Dateien zur Verfügung zu stellen, da diese mit Hilfe des Kommandos Archiv extrahieren leicht von ihm entpackt werden können.

Hinweis: Die Archivierungsfunktion ist nicht zur Speicherung eines Projekts gedacht, vielmehr zum einfachen Zusammenfassen aller projektrelevanten Dateien.

Nach Aufruf des Kommandos öffnet sich der Dialog **Projektarchiv**:

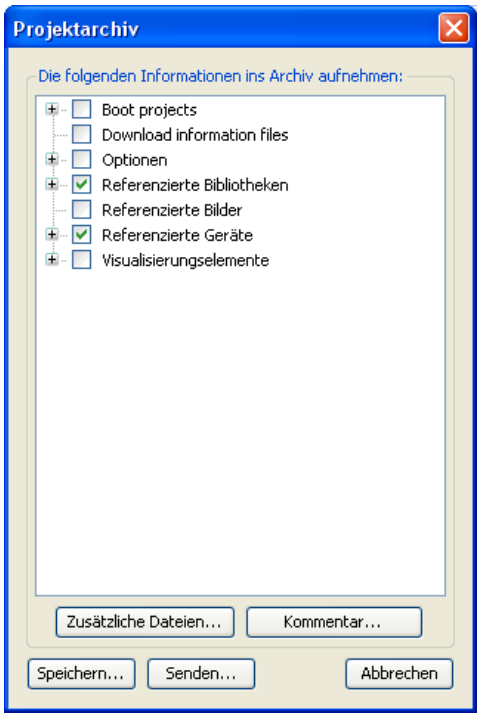


Abb. 137 Projektarchiv, Dialog

Innerhalb dieses Fensters bestimmt man, welche Dateikategorien zum Archiv hinzugefügt werden sollen. Durch Aktivieren oder Deaktivieren des entsprechenden Kontrollkästchens wird eine Kategorie ausgewählt bzw. aus der Auswahl entfernt. Aktivieren (bzw. Deaktivieren) erfolgen durch einen einfachen Klick in das Kontrollkästchens oder aber durch einen Doppelklick auf den Namen der Kategorie. Ist eine Kategorie mit einem Haken versehen ☑, so werden alle zu dieser Kategorie gehörenden Dateien dem Archiv hinzugefügt. Enthält das aktuelle Projekt keinerlei Dateien einer bestimmten Kategorie, so ist diese automatisch deaktiviert ☐.

Um eine einzelne Datei einer bestimmten Kategorie selektieren bzw. zu deselektieren, klicken Sie zunächst auf das der Kategorie vorangestellte Zeichen ⊞, welches eine Liste aller zugehörigen Dateien öffnet. Standardgemäß sind alle zu einer aktivierten Kategorie gehörenden Dateien ausgewählt. Um eine Auswahl zu ändern, selektieren oder deselektieren Sie die entsprechende Datei in der selben Weise wie Kategorien. Falls nun also nicht alle, sondern nur Teile der zu einer Kategorie gehörenden Dateien selektiert sind, so wird diese Kategorie automatisch durch ☑gekennzeichnet.

Um dem Archiv andere als die oben angeführten Dateien hinzuzufügen, benutzen Sie die Schaltfläche **Zusätzliche Dateien...** . Hierauf öffnet folgender Dialog:

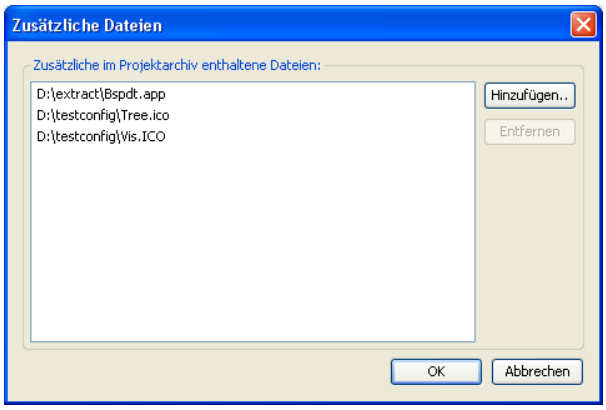


Abb. 138 Zusätzliche Dateien, Dialog

Die Schaltfläche **Hinzufügen** öffnet den Standarddialog zur Suche nach Dateien, die hierin ihrem Typ gemäß angeordnet sind. Nach Bestätigung der Auswahl mit **Öffnen** werden die selektierten Dateien zur Liste des **Zusätzliche Dateien...** Dialogs hinzugefügt. Um eine Datei aus dieser Liste zu entfernen, selektieren Sie diese durch Anklicken und verwenden Sie anschließend die Schaltfläche **Entfernen**. Sobald die Liste der ausgewählten Dateien Ihren Anforderungen entspricht, können Sie den Dialog mit **OK** schließen.

Um dem Archiv ein Kommentar beizufügen, drücken Sie die Taste **Kommentar...**. Ein Text-Editor zur Eingabe des Kommentars erscheint. Schließen Sie den Editor nach Beenden der Eingabe mit **OK**. Bei Import des Projektarchivs kann Ihr Kommentar mittels der Funktion **Kommentar anzeigen...** eingesehen werden.

Nachdem Sie alle gewünschten Einstellung vorgenommen haben, betätigen Sie einer der folgenden drei Tasten des Hauptdialogs:

- **Speichern...** erzeugt die Archivdatei und speichert sie in einem Verzeichnis, das Sie mittels des aufgehenden Dialogfensters spezifizieren. In letzterem kann auch der standardmäßige Name der Archivdatei abgeändert werden. Eine anschließende Bestätigung mit **Speichern** startet den Archivierungsprozess.
- **Senden...** erzeugt eine temporäre Archivdatei, die einer simultan erzeugten leeren Email als Anlage beigefügt wird. Die erfolgreiche Ausführung dieser Operation setzt eine korrekte Installation des MAPI (Messaging Application Programming Interface) voraus. Ein Fehlschlagen wird durch Ausgabe einer entsprechenden Fehlermeldung dokumentiert. Das temporäre Archiv wird automatisch nach Senden der Email gelöscht.
- **Abbrechen** um die Aktion zu verwerfen; es wird kein Projektarchiv erzeugt.

Projektarchiv: Archiv extrahieren...

Dieser Menüpunkt (Kategorie 'Datei') dient zum Extrahieren eines Projektarchivs, das mit Hilfe des Befehls Projektarchiv: Archiv speichern/versenden... angelegt wurde und standardmäßig die Dateiendung ".projectarchive" trägt.

Zu beachten ist, dass vor dem Extrahieren eines Archivs verlangt wird, dass **alle gerade geöffneten Projekte** in allen gerade laufenden Instanzen des Programmiersystems **geschlossen** werden. Dies dient dazu eine direkte Auswirkung der Extraktion auf laufende Projekte zu vermeiden, beispielsweise durch einen Wechsel der Bibliotheksdateien.

Auf den Aufruf des Menüpunktes hin öffnet sich ein Standarddialog zur Auswahl des gewünschten Archivs. Nach Bestätigung der Auswahl mit **Öffnen** erscheint folgendes Dialogfenster:

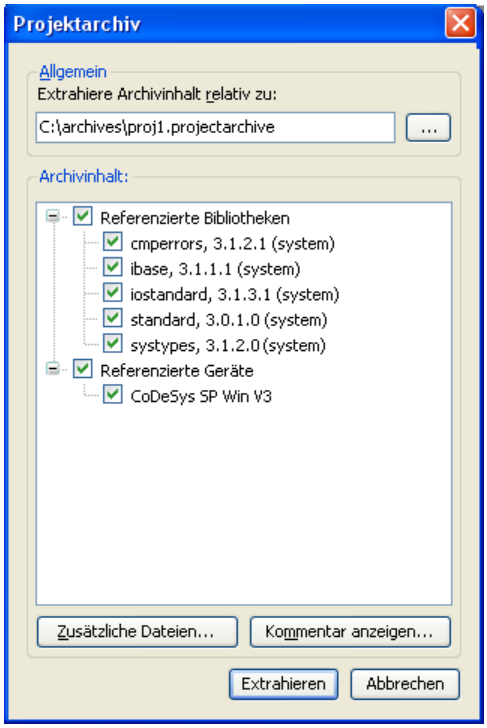


Abb. 139 Projektarchiv, Dialog

General: in **Extrahiere Archivinhalt relativ zu:** wird das Verzeichnis angezeigt, in welches das Archiv entpackt wird. Zur Abänderung können Sie den Pfad überschreiben oder das rechts stehende Symbol ... anklicken um das Detaiverzeichnis mit Hilfe des Standarddialogs zu durchsuchen.

Archivinhalt: In diesem Fenster sind alle im Archiv enthaltenen Dateikategorien verzeichnet. Ein Klick auf das jeder Kategorie vorangestellte Zeichen + erweitert die Darstellung um eine Liste aller zur Kategorie gehörenden Dateien. Standardgemäß sind alle aufgeführten Kategorien mit ☑ gekennzeichnet, weshalb alle der jeweils zugehörigen Dateien entpackt werden. Falls die das Extrahieren bestimmter Dateien oder sogar ganzer Kategorien unterbinden möchten, deaktivieren Sie Ihre Kennzeichnung durch einen Mausklick. Die Markierung wird daraufhin abgeändert in ■ bei partieller Abwahl, beziehungsweise in ☐ für gesamte Abwahl.

Zusätzlich zu den Projektdateien kann das Archiv weitere Dateien enthalten, die vom Ersteller absichtlich hinzugefügt wurden. Nach einem Klick auf **Zusätzliche Dateien...** erscheint ein Dialogfenster:

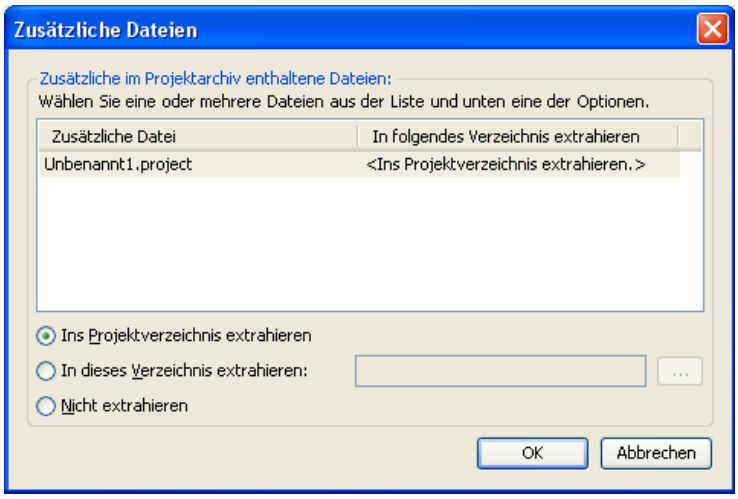


Abb. 140 Zusätzliche Dateien, Dialog

welches eine komplette Liste der beigefügten Dateien enthält. Standardgemäß werden diese zusätzlichen Dateien im Zuge des Extrahierens nicht entpackt (vgl. den Hinweis **<Nicht extrahieren>** rechts der Dateinamen). Sollten Sie dies wünschen, so markieren Sie die entsprechenden Zusatzdateien durch Anklicken ihres Namens und wählen unter einer der drei angebotenen Möglichkeiten:

- **Ins Projektverzeichnis extrahieren:** die ausgewählten Zusatzdateien werden in dasselbe Verzeichnis (s. spezifizierter path) entpackt wie die Projektdateien. Der Eintrag rechts des Dateinamens wird abgeändert in **<Ins Projektverzeichnis extrahieren>**.
- **In dieses Verzeichnis extrahieren:** die ausgewählten Zusatzdateien werden in ein Verzeichnis gelegt, das entweder durch Angabe des entsprechenden Pfades ausgewählt werden kann oder mit Hilfe des Standarddialogs, der sich nach einem Klick auf ... öffnet. Der Eintrag rechts des Dateinamens enthält hierauf den angegebenen Pfad.
- **Nicht extrahieren:** löscht Dateien aus der Auswahl der zu extrahierenden Zusatzdateien.

Nach Anpassung der Optionen kann der Dialog 'Zusätzliche Dateien' mit **OK** verlassen werden.

Über Schaltfläche **Kommentar anzeigen...** hin wird ein eventuell vom Projekt-Ersteller hinzugefügter Kommentar angezeigt. Sollte ein solcher Kommentar nicht vorhanden sein, wird stattdessen eine entsprechende Fehlermeldung angezeigt.

Nach Abschluss der Konfiguration im Dialog-Fenster wählen Sie

- **Extrahieren** um das Archiv in das angegebene Zielverzeichnis zu entpacken.

  Sofern im Zielverzeichnis schon eine Datei existiert, die denselben Namen trägt wie eine im Projektarchiv enthaltene Datei, so wird ein Warnhinweis ausgegeben und Sie werden gefragt werden, ob die lokale Datei ersetzt werden soll. Sie können Ihre Entscheidung für jeden weiteren Namenskonflikt gültig machen indem Sie die Option ☑ **Auf alle Objekte und Dateien anwenden** aktivieren.

- **Abbrechen** um die Aktion abzubrechen und ein Extrahieren des Projektarchivs zu verhindern.

Projekt kopieren

DIESE FUNKTION IST DERZEIT LEIDER NICHT VERFÜGBAR.

Dieser Befehl (Kategorie 'Datei') dient dem Zusammenführen zweier »PLC Designer« Projekte, d.h. dem Kopieren eines Projekts in ein anderes. Standardmäßig ist der Befehl nicht im Datei-Menü, sondern im Projekt-Menü enthalten.

'Projekt kopieren'-Dialog:

Der Befehl öffnet den Dialog 'Projekt kopieren'. Es handelt sich um den Windows Standard-Browser, mit dem Sie das Projekt auswählen, das in das aktuelle hinein kopiert werden soll.

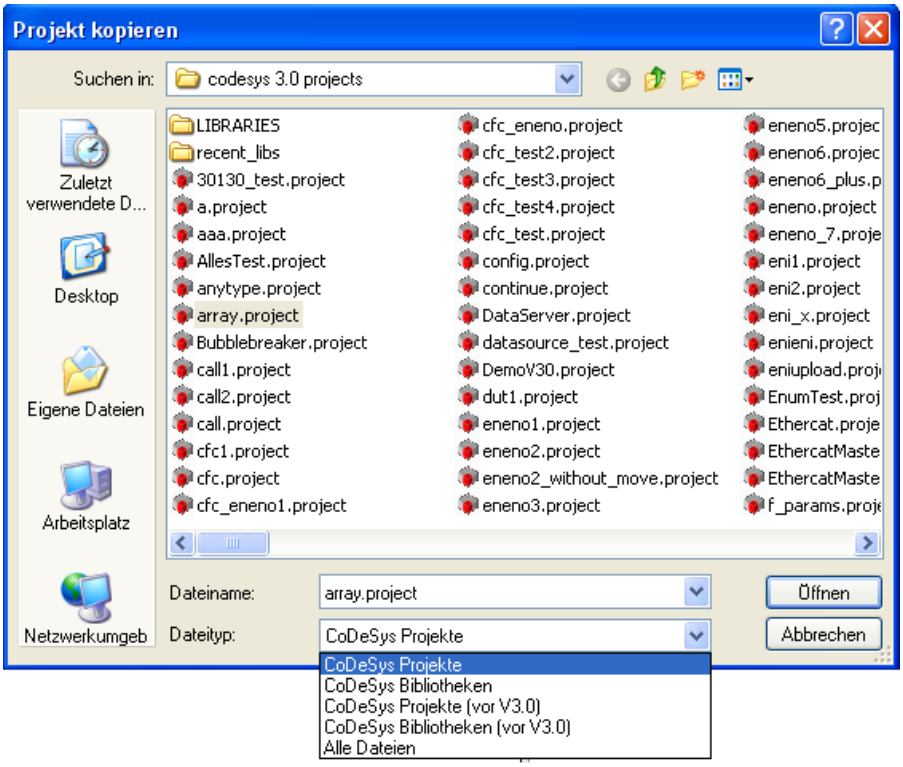


Abb. 141 Dialog 'Projekt kopieren'

Der Filter für den **Dateityp** ist standardmäßig auf "»PLC Designer« Projekte" (Erweiterung ".project") gesetzt. Dies sind Projekte, die mit einer »PLC Designer«-Version nach V2.3 erzeugt wurden. Das gleiche gilt für den Typ "»PLC Designer« Bibliotheken". Welche weiteren Dateitypen angeboten werden, hängt davon ab, welche Konverter installiert sind.

Wählen Sie die gewünschte Projektdatei und bestätigen mit **Öffnen**.

Wenn das zu kopierende Projekt ein anderes Format als das bereits geöffnete hat (anderer Dateityp) und zuerst konvertiert werden muss, erhält der Benutzer die entsprechenden Dialoge, wie beim Öffnen eines Projekts anderen Formats.

Dialog 'Projekt kopieren-<projektname>':

Nach der Auswahl des zu kopierenden Projekts und einer eventuellen Konvertierung wird der Dialog 'Projekt kopieren -<projektname>' geöffnet, wo im einzelnen festgelegt wird, welche Objekte des Projekts in das aktuell bearbeitete kopiert werden sollen:

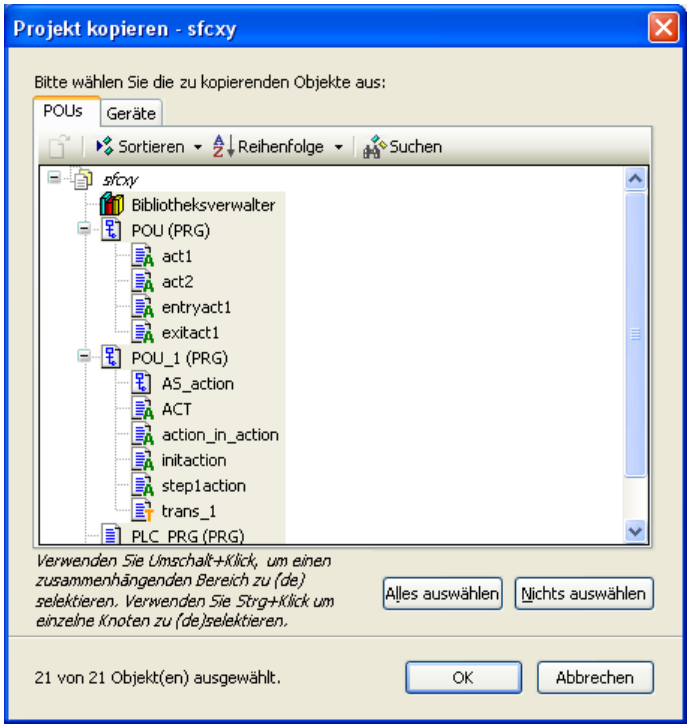


Abb. 142 Dialog 'Projekt kopieren', Objekte auswählen

Die Objekte des Projekts werden in Registerblättern angezeigt, die den im Projekt verwendeten Objektbäumen entsprechen, im Standardfall also "POUs" und "Geräte". Baumstruktur und Navigationsmöglichkeiten (Symbole oberhalb der Auswahllisten) entsprechen ebenfalls denen im jeweiligen Fenster.

Wählen Sie die gewünschten Objekte aus. Eine Mehrfachauswahl ist bei gedrückter <Umsch>-Taste möglich, wobei mit <Strg>+ Mausklick eine einzelne Auswahl auch wieder aufgehoben werden kann. Die Schaltflächen **Alles auswählen** bzw. **Nichts auswählen** können verwendet werden, um jeweils den gesamten Baum auszuwählen bzw. eine Auswahl insgesamt aufzuheben.

Mit **OK** wird der Auswahldialog geschlossen und der Kopierprozeß gestartet

Wenn festgestellt wird, dass im Projekt bereits Objekte mit gleichen Namen wie die zu kopierenden vorliegen, erscheint der folgende Dialog:

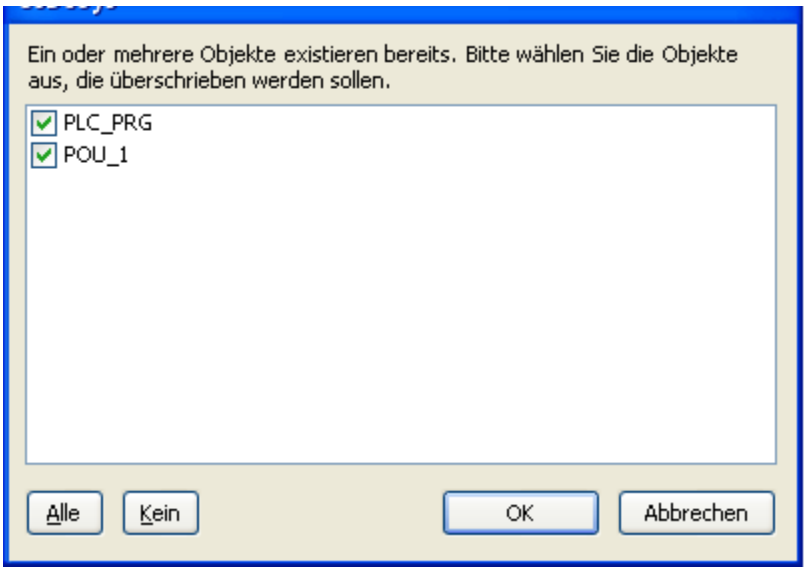


Abb. 143 Dialog für das Auswählen von Objekten, die überschrieben werden sollen

Hier werden alle Objekte angezeigt, die ins aktuelle Projekt kopiert werden sollen, aber für die hier bereits Objekte mit gleichem Namen vorliegen.

Die mit einem Haken versehenen Objekte werden kopiert und überschreiben die bereits vorhandenen. Wenn der Haken entfernt wird, wird das Objekt zwar kopiert, aber automatisch mit einem eindeutigen Namen versehen so dass das vorhandene Objekt erhalten bleibt (üblicherweise wird ein "_<n>" an den Objektnamen gehängt, wobei n eine fortlaufende Nummer ist).

Alle versieht alle aufgelisteten Einträge mit einem Haken, **Kein** entfernt alle Haken.

Mit **OK** wird der Kopierprozeß fortgesetzt und die kopierten Objekte in den Objektbäumen des aktuellen Projekts eingefügt.

Zuletzt verwendete Projekte...

Dieser Befehl (Kategorie 'Datei'), üblicherweise im Datei-Menü, öffnet eine Liste der zuletzt geöffneten Projekte, von der eines ausgewählt und damit wieder geöffnet werden kann. Die Anzahl der Projekte, die in dieser Liste gehalten werden, wird im Optionen-Dialog Kategorie 'Laden und Speichern' definiert.

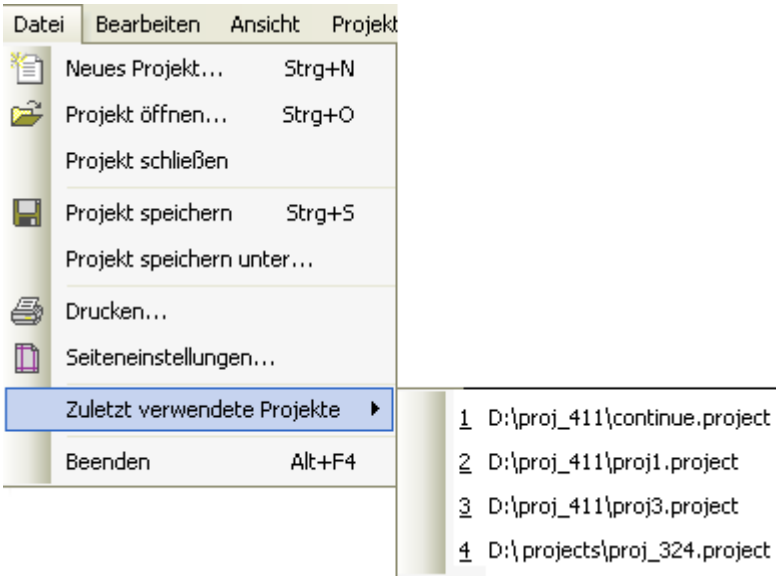


Abb. 144 Liste zuletzt verwendeter Projekte

Beenden

Shortcut <Alt>+<F4>

Der Befehl (Kategorie 'Datei') beendet das Programmiersystem. Wenn gerade ein Projekt geöffnet ist, das seit dem letzten Speichern verändert wurde, erscheint ein Dialog, in dem Sie gefragt werden, ob das Projekt gespeichert werden soll.

Vergleichen...

Symbol:

Mit Hilfe des Kommandos **Projekt / Vergleichen** (Kategorie 'Datei') kann das aktuelle Projekt mit einem anderen, dem sog. Referenzprojekt verglichen werden. Das Referenzprojekt ebenso wie die Vergleichsoptionen werden spezifiziert im Dialog 'Projektvergleich', welcher bei Aufruf des Kommandos geöffnet wird.

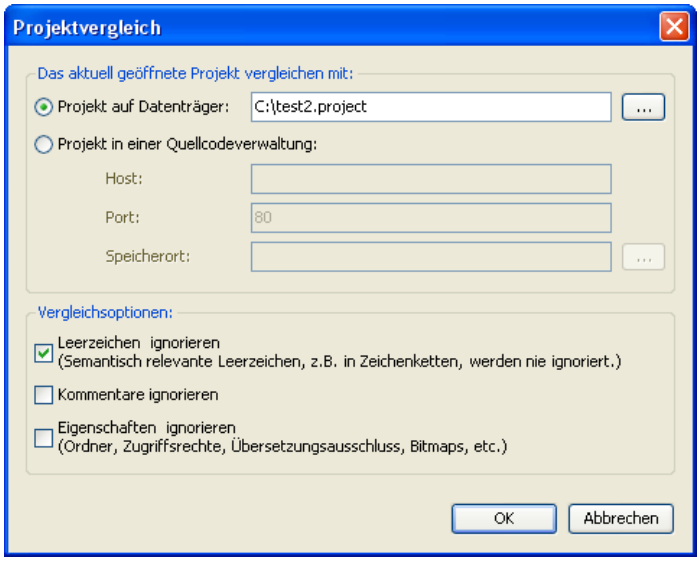


Abb. 145 Dialog Projektvergleich

Das aktuell geöffnete Projekt vergleichen mit:

Das Referenzprojekt ist

- entweder ein **Projekt auf Datenträger**: standardmäßig ist der Pfad des aktuelle Projekts eingetragen, was zur Folge hat, dass dieses mit seiner zuletzt gespeicherten Version verglichen wird. Nach einem Klick in das Textfeld kann der Pfad und damit das Referenzprojekt abgeändert werden. Alternativ kann es durch Zuhilfenahme des Standarddialogs zur Dateisuche ersetzt werden, welcher mittels der Taste ... geöffnet wird.
- oder ein **Projekt in einer Quellcodeverwaltung**: hierbei muss zunächst der Name des Host und die Nummer des entsprechenden Ports angegeben werden. Anschließend kann der Pfad des gewünschten Referenzprojekts entweder direkt im Textfeld eingetippt werden oder aber ausgewählt werden im Standarddialog zur Dateisuche, welcher nach einem Klick auf ... zur Verfügung steht.

Vergleichsoptionen:

Durch Anklicken können hier folgende Optionen betreffs des Vergleichs aktiviert werden:

- **Leerzeichen ignorieren**: Abweichungen des aktuellen vom Referenzprojekt, welche allein auf Leerzeichen basieren, bleiben unerwähnt.
- **Kommentare ignorieren**: Kommentare werden vom Vergleich ausgeschlossen.
- **Eigenschaften ignorieren**: Eigenschaften des Objekts werden nicht verglichen.

Nach Schließen des Dialogs 'Projektvergleich' mit **OK** wird der Vergleich gemäß der getroffenen Einstellungen ausgeführt.

Erfassung des Ergebnisses eines Vergleichs mittels eines markierten Gerätebaums

Das Ergebnis des Vergleichs wird in einem neuen Fenster mit dem Titel "**Projektvergleich - Unterschiede**" dargestellt.

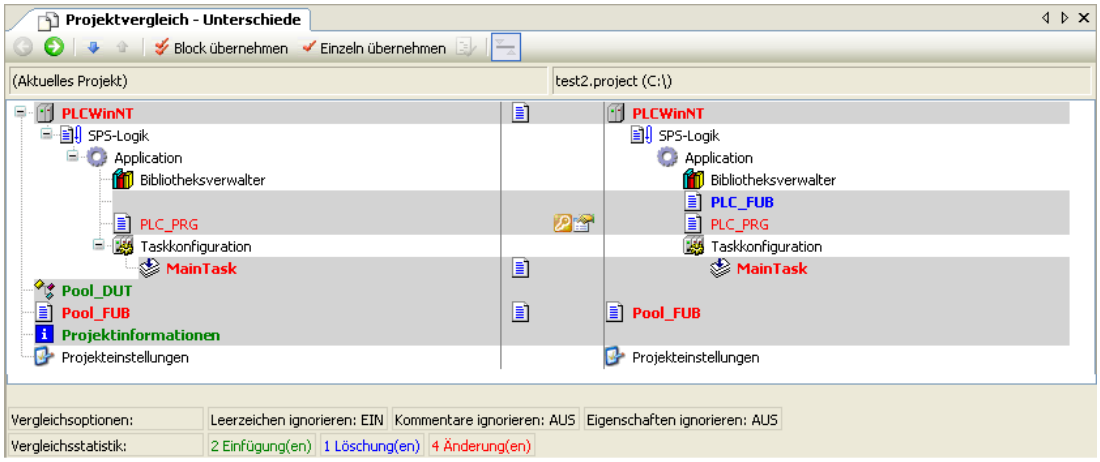


Abb. 146 Beispiel eines Projektvergleichs

Den Kopf des neuen Fensters bildet eine Symbolleiste (s. u. "Arbeiten im Vergleichsmodus") gefolgt von einer Titelleiste und einem Unterfenster. Die Titelleiste wie auch das Unterfenster selbst sind unterteilt: die linke Hälfte gehört zum aktuellen Projekt, die rechte zum Referenzprojekt. Die entsprechenden Dateipfade werden in der Titelleiste angezeigt, wohingegen das Unterfenster die entsprechenden Gerätebäume abbildet.

In diesen sind identische Teile in schwarzer Schrift ohne weitere Zusätze vermerkt. Im Fall von Abweichung steht der Name des betroffenen Elements

- in **blauem Fettdruck**, falls das Element nur im Referenzprojekt vorhanden ist; anstelle seines Namens ist an der entsprechenden Stelle im Gerätebaum des aktuellen Projekts eine Lücke zu sehen.
- in **grünem Fettdruck**, falls das Element nur im aktuellen Projekt vorhanden ist; anstelle seines Namens ist an der entsprechenden Stelle im Gerätebaum des Referenzprojekts eine Lücke zu sehen.

- in **rotem Fettdruck** in beiden Teilen des Unterfensters, falls sich die beiden Versionen des Elements hinsichtlich der Implementierung unterscheiden. In diesem Fall ist dem Namen des Elements zusätzlich das Symbol nachgestellt.
- in roter Schrift, falls die Unterschiede der beiden Versionen nur die Eigenschaften des Elements betreffen (Anzeige durch nachgestelltes Symbol), die Zugriffsrechte (angezeigt durch nachgestelltes Symbol) oder aber hinzugefügte Ordner (angezeigt durch nachgestelltes Symbol). Im letzten Fall wird der entsprechende Pfad des betroffenen Elements direkt unterhalb des Unterfensters angezeigt, sobald Sie den Mauszeiger auf den Elementnamen (im aktuellen oder aber im Referenzprojekt) plazieren.

Zusätzlich wird jede Zeile, welche einen farbigen Eintrag enthält, grau hinterlegt.

Unterhalb des Unterfensters befinden sich weitere Zeilen, welche die für den Vergleich gewählten Optionen anzeigen ebenso wie eine knappe Statistik, welche die Anzahl der dem aktuellen Projekt neu hinzugefügten und gelöschten Elemente sowie die Zahl der Abänderungen enthält.

Arbeiten im Vergleichsmodus

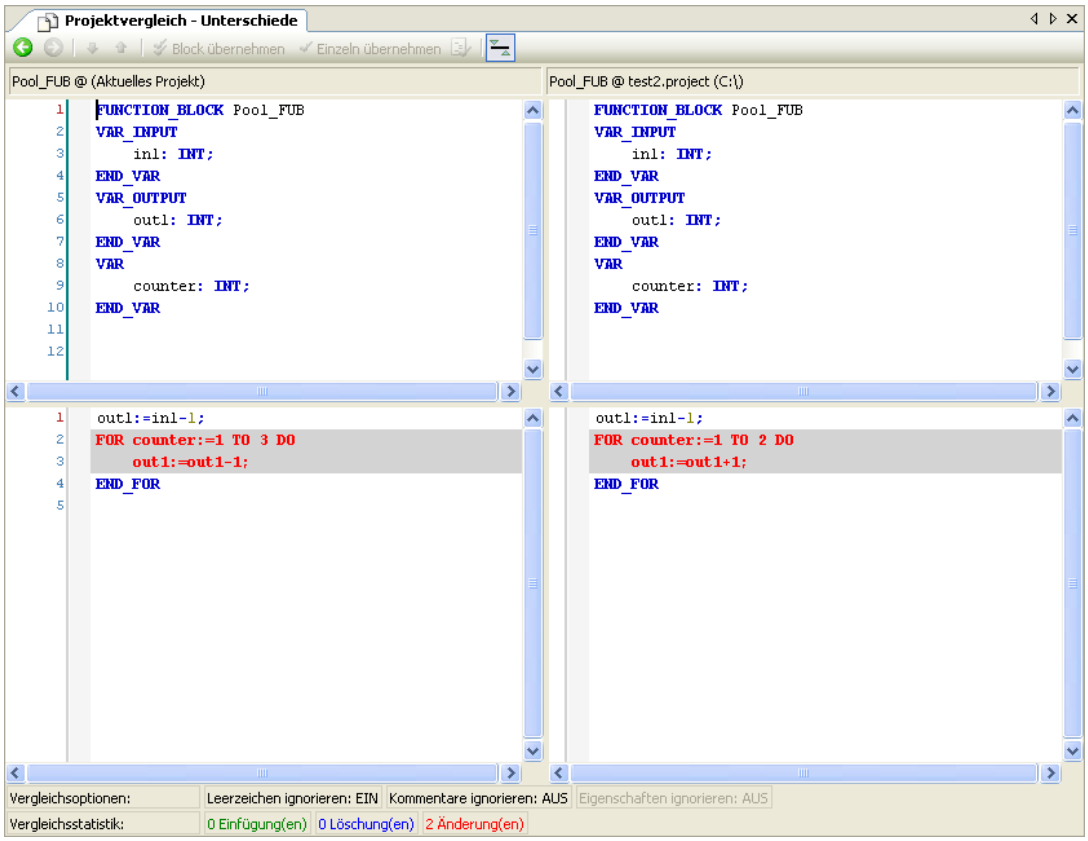
Die Symbolleiste im oberen Teil des Vergleichsfensters stellt folgende Funktionen zur Bearbeitung des Vergleichs bereit:

- Um detaillierte Informationen zu den Unterschieden in der Implementation eines in rotem Fettdruck geschriebenen Elementes zu erhalten, führen Sie einen Doppelklick auf dessen Namen aus oder wählen Sie aus der Symbolleiste. Hierauf wird die Vergleichsübersicht von einer detaillierten Ansicht überlagert, die den Inhalt der beiden Versionen gegenüberstellt. Dabei wird ein der obigen Beschreibung entsprechendes Farbschema verwendet.
  Im Fall eines Funktionsbausteins wird dessen Implementierung im entsprechenden Graphik- oder Text-Editor dargestellt, **links für das aktuelle Projekt** und **rechts für das Referenzprojekt**. Die kleinsten Einheiten, die miteinander verglichen werden, sind Zeilen (ST, IL), Netzwerke (FBD, LD) oder Elemente (CFC, SFC). Im Fall eines Gerätes werden veränderbare Einheiten wie Adresse, Parameter oder die Abbildungsliste tabellarisch verglichen. Wie schon oben erwähnt, befindet sich in der linken Seite der Tabelle die Einträge des aktuellen Projekts und in der rechten Seite die Einträge des referenzierten Gerätes. Beachten Sie, dass Sie keine der beiden Versionen innerhalb dieser Gegenüberstellung bearbeiten können. Es ist nur möglich anstelle der aktuellen Version die Version des Referenzprojektes für das aktuelle Projekt zu akzeptieren (s. Punkt 5 und 6).
- Wählen Sie ganz links in der Symbolleiste um zur Vergleichsübersicht zurückzukehren.
- Auf Anklicken des Symbols hin wird die nächste Unterschiede aufweisende Einheit des Gerätebaums selektiert.
- Auf Anklicken des Symbols hin wird die vorausgehende Unterschiede aufweisende Einheit des Gerätebaums selektiert.

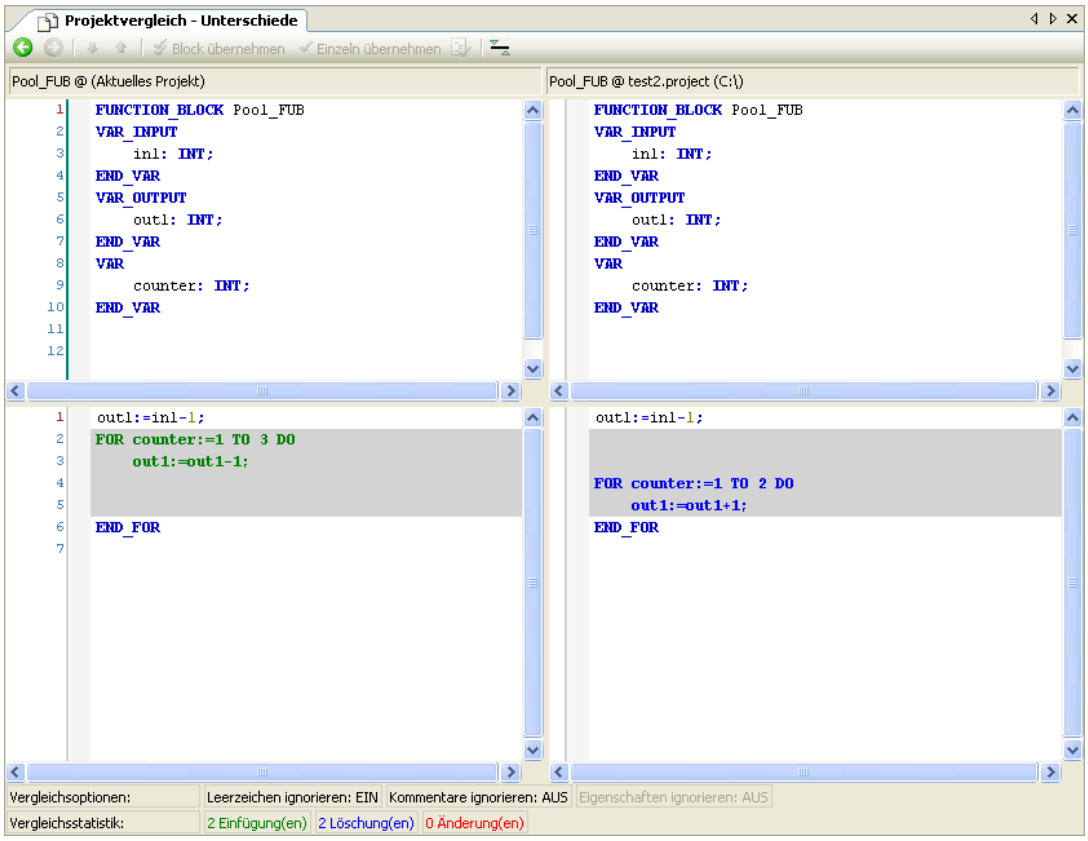
- Bei Auswahl von Accept Block wird ein **Block** in der Version, wie er im Referenzprojekt enthalten ist, auch für das aktuelle Projekt akzeptiert. Die **Akzeptanz ist ausschließlich in dieser Richtung möglich**.
  In der detaillierten Ansicht besteht ein Block aus der Einheit (Zeile, Netzwerk, Element), in welcher der Cursor aktuell plaziert, und aus allen zugehörigen Einheiten, welche dieselbe Markierung von Unterschieden aufweisen (z. B. Folgezeilen). Nach Akzeptanz eines Blockes innerhalb der detaillierten Ansicht wird dieser im aktuellen Projekt gelb hervorgehoben.
  In der Vergleichsübersicht hingegen besteht ein Block aus der aktuell selektierten Einheit zusammen mit allen unterstellten Einheiten. Nach Akzeptanz eines Blockes innerhalb der Übersicht werden die zugehörigen Einheiten gelb hinterlegt und mit einem Häkchen versehen. Die Akzeptanz eines Blockes, der nur in der aktuellen Version vorhanden ist, wird ihn dort löschen (was durch ein zusätzliches kleines rotes Kreuz in der nun leeren gelb hinterlegten Zeile angezeigt wird). Ein in der aktuellen Version gelöschter Block hingegen wird bei Akzeptanz wieder eingefügt (und durch ein zusätzliches kleines grünes Kreuz gekennzeichnet). Einzelne Teile eines Blockes können mit Hilfe des Kommandos Accept Singleübernommen bzw. gelöscht werden. Wird nach einer Akzeptanz die detaillierte Ansicht der betroffenen Einheit geöffnet (siehe oben, erster Punkt der Liste), so wird gefragt, ob die Änderung auch für diese Ansicht übernommen werden sollen.
  Ein wiederholter Klick auf Accept Block macht die Effekte seiner letzten Anwendung rückgängig.
- Bei Auswahl von Accept Single wir die aktuell selektierte Einheit für das aktuelle Projekt in der Version akzeptiert, die im Referenzprojekt enthalten ist. Die **Akzeptanz ist ausschließlich in dieser Richtung möglich**.
  Innerhalb der detaillierten Ansicht wird eine akzeptierte Einheit in der aktuellen Version gelb hinterlegt. Bei Rückkehr zur Vergleichsübersicht werden Sie gefragt, ob die Änderungen übernommen werden sollen. Sind beide Versionen daraufhin identisch, so werden sie in der Übersicht nun in regulärer schwarzer Schrift angezeigt.
  Nach Akzeptanz einer einzelnen Einheit innerhalb der Vergleichsübersicht hingegen wird sowohl ihr Name gelb hinterlegt als auch der Name aller Einheiten, von welchen sie abhängt. Zusätzlich wird den Namen ein Häkchen angefügt. Die Akzeptanz einer im aktuellen Projekt neu eingefügten Einheit wird diese selbst wie auch alle von ihr abhängenden Einheiten löschen und die entsprechenden Zeilen mit einem kleinen roten Kreuz markieren. Die Akzeptanz einer im aktuellen Projekt gelöschten Einheit wird diese selbst wieder in das aktuelle Projekt einfügen wie auch alle Einheiten, von denen sie abhängt. Ihre Name werden zusätzlich durch ein kleines grünes Kreuz gekennzeichnet. Wird nach einer Akzeptanz die detaillierte Ansicht der betroffenen Einheit geöffnet (siehe oben, erster Punkt der Liste), so wird gefragt, ob die Änderung auch für diese Ansicht übernommen werden sollen.
  Ein wiederholter Klick auf Accept Singlemacht die Effekte seiner letzten Anwendung rückgängig.

- Mit Hilfe von wird d**er Ordner, die Zugriffsrechte oder Objekteigenschaften der selektierten Einheit** in der im Referenzprojekt enthaltenen Version für das aktuelle Projekt akzeptiert. Die **Akzeptanz kann ausschließlich in dieser Richtung vorgenommen werden**. Die gewünschte Auswahl aus Ordner, Zugriffsrechten und Objekteigenschaften kann in dem eingeblendeten Dialogfeld getroffen werden. Die betroffenen Einheiten werden in Folge gelb unterlegt; den Symbolen, welche Ordner, Zugriffsrechte bzw. Objekteigenschaften repräsentieren, werden entsprechend Häkchen zugefügt.
- Das letzte Symbol der Werkzeugleiste kann nur **innerhalb der detaillierten Ansicht** betätigt werden. Standardgemäß werden voneinander abweichende Einheiten (Zeilen, Netzwerke, Elemente) in roter Farbe einander gegenübergestellt. Durch einen Klick auf wird diese **Darstellung folgendermaßen abgeändert**: die voneinander abweichenden Einheiten werden im aktuellen Projekt nun angezeigt wie neu eingefügte Einheiten (**grüner Fettdruck**), im Referenzprojekt hingegen wie im aktuellen Projekt gelöschte Einheiten (**blauer Fettdruck**). Beachten Sie, dass eine Abweichung, die in der Statistik vormals als Abänderung gezählt wurde nun als Einfügung und als Löschung gezählt wird. Ein wiederholtes Betätigen von stellt die standardmäßige Darstellung wieder her.

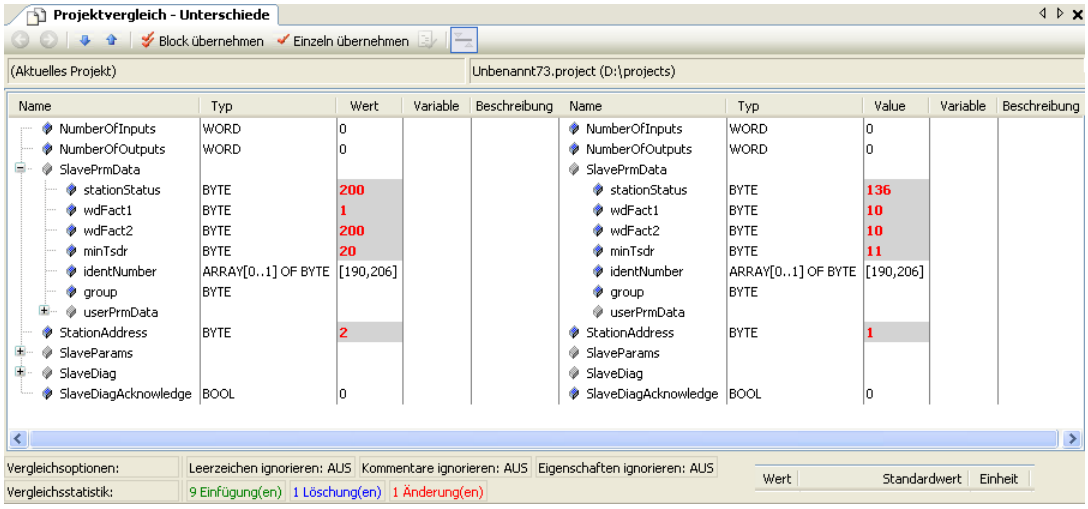
Beispiel eines detaillierten Vergleichs zweier Versionen eines FUB (ST) in Standarddarstellung:



Beispiel eines detaillierten Vergleichs zweier Versionen eines FUB (ST) in nicht gegenüberstellender Darstellung:



Beispiel eines detaillierten Vergleichs zweier Versionen eines Gerätes:



Projektumgebung

Dieser Befehl (Kategorie 'Datei') öffnet den Dialog **Projektumgebung**, in dem Einstellungen zur Versionskompatibilität des Projekts liegen.

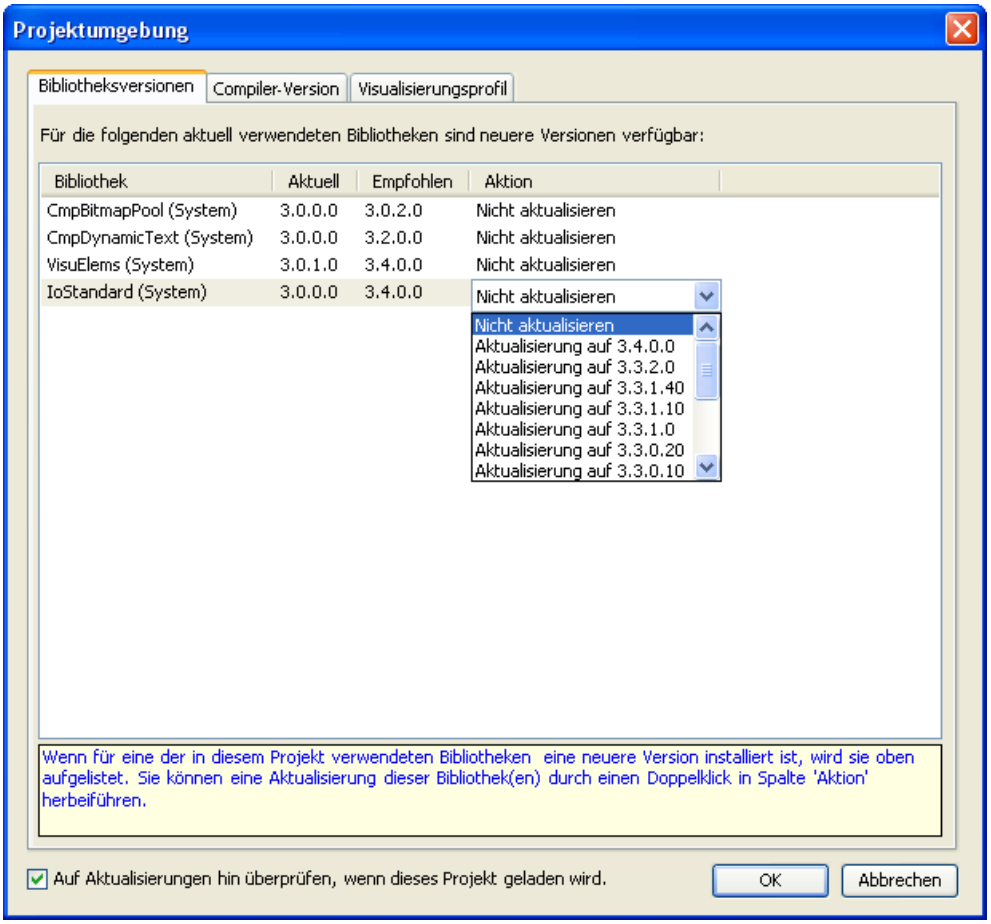


Abb. 147 Dialog Projektumgebung, Beispiel

Dieser Dialog erscheint automatisch, wenn ein Projekt geöffnet wird, das mit anderen Bibliotheksversionen oder anderen Compiler- und Visualisierungsprofil-Version(en) gespeichert worden war. Wenn die Option **Auf Aktualisierungen hin überprüfen, wenn dieses Projekt geladen wird** aktiviert ist, wird das bei jedem Öffnen dieses Projekts geschehen, ansonsten nur beim ersten Mal.

Siehe folgende Unterdialoge:

Bibliotheksversionen

Alle Bibliotheken, die im Projekt explizit eingebunden wurden (implizit eingebundene werden im Bibliotheksverwalter grau dargestellt), und die nicht in der neuesten verfügbaren Version verwendet werden, werden mit ihrer **Aktuell**en und **Empfohlen**en Versionsnummer aufgelistet. Wählen Sie die gewünschte Aktualisierungsaktion (Standardeinstellung: Nicht aktualisieren) aus der Auswahlliste, die sich bei einem Doppelklick auf das Feld **Aktion** öffnet.

Compile-Version

Die **Aktuell**e und **Empfohlen**e Version werden angezeigt und man kann eine Aktualisierungsaktion ((Standardeinstellung: Nicht aktualisieren) aus der Auswahlliste auswählen, die sich bei einem Doppelklick auf das Feld **Aktion** öffnet.

Hinweis: Wenn ein Projekt geöffnet wird, das mit einer älteren Version des Programmiersystems erstellt wurde, und für das die Compiler-Version "Neueste.." ist (Projekteinstellungen), während hier im neuen Programmiersystem die Projektumgebungseinstellung für die Compile-Version "Nicht aktualisieren" ist, dann wird die zuletzt im alten Projekt verwendete Compile-Version weiter verwendet (nicht die "Aktuelle" in der neuen Umgebung).

Visualisierungsprofil

Die **Aktuell**e und **Empfohlen**e Version werden angezeigt und man kann eine Aktualisierungsaktion ((Standardeinstellung: Nicht aktualisieren) aus der Auswahlliste auswählen, die sich bei einem Doppelklick auf das Feld **Aktion** öffnet.

OK startet die gerade definierten Aktualisierungsaktionen.

Projektinformationen

Symbol:

Dieser Befehl gehört zur Befehlskategorie 'Datei', ist aber standardmäßig nicht im Datei- sondern im Projekt-Menü zu finden.

Er öffnet den Dialog 'Projektinformationen', in dem die Eigenschaften des Projekts und zusätzliche Informationen (z.B. Zugriffsrechte, Versionsnummer, Autor, Firma, Statistik zu den Projektobjekten etc.) angezeigt bzw. definiert werden können. Dieser Dialog ist in einem "Standardprojekt" bereits als Objekt im POUs-Fenster eingefügt.

Beachten Sie die Möglichkeit, mit Hilfe von Eigenschaften-Schlüsseln und zusätzlich angelegten Funktionen von extern auf die Projektinformationen zuzugreifen (s.u.).

Vier Registerblätter sind für die Informationen-Kategorien 'Datei', 'Zusammenfassung', 'Eigenschaften' und 'Statistiken' verfügbar:

1. Datei

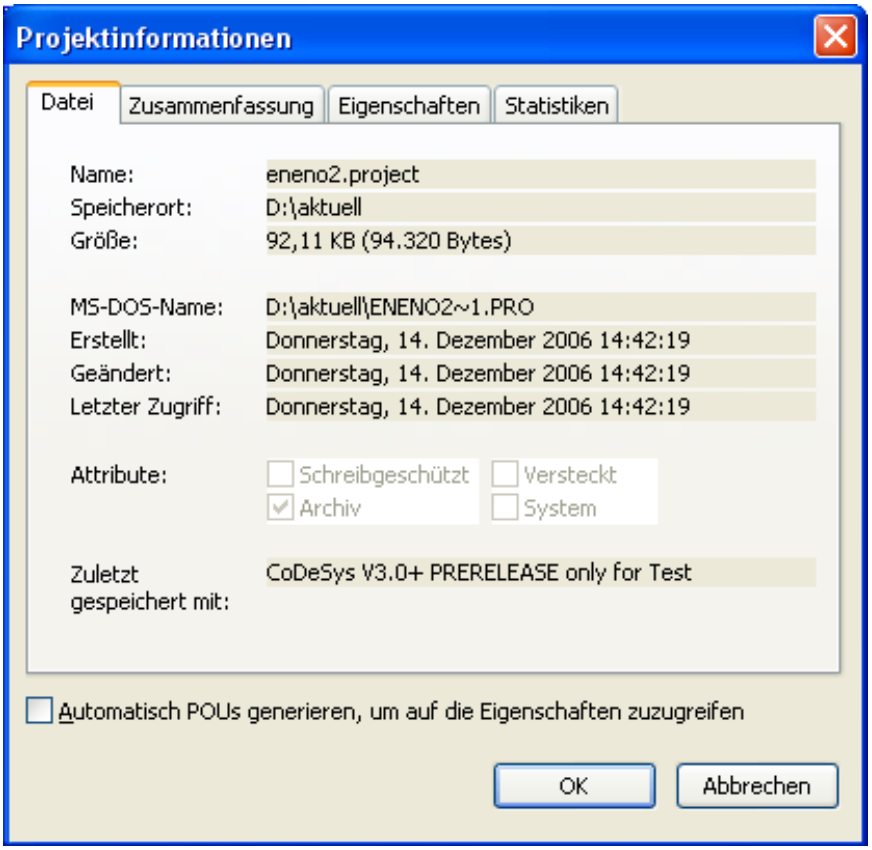


Abb. 148 Projektinformationen Dialog, Datei

Hier werden die folgenden Eigenschaften einer Projektdatei angezeigt: Name, Speicherort, Größe in KBytes, MS-DOS-Name, Erstelldatum, Datum des letzen Zugriffs und Name des Profils, mit dem die Datei zuletzt gespeichert wurde.

Außerdem sieht man, welche der folgenden Dateiattribute gerade gesetzt sind: Schreibgeschützt, Versteckt (d.h. per Default nicht im Explorer sichtbar), Archiv (bereit für Archivierung), System (Systemdatei). Diese Attribute sind standardmäßig hier nicht editierbar (siehe Dateieigenschaften im Windows Explorer)..

2. Zusammenfassung

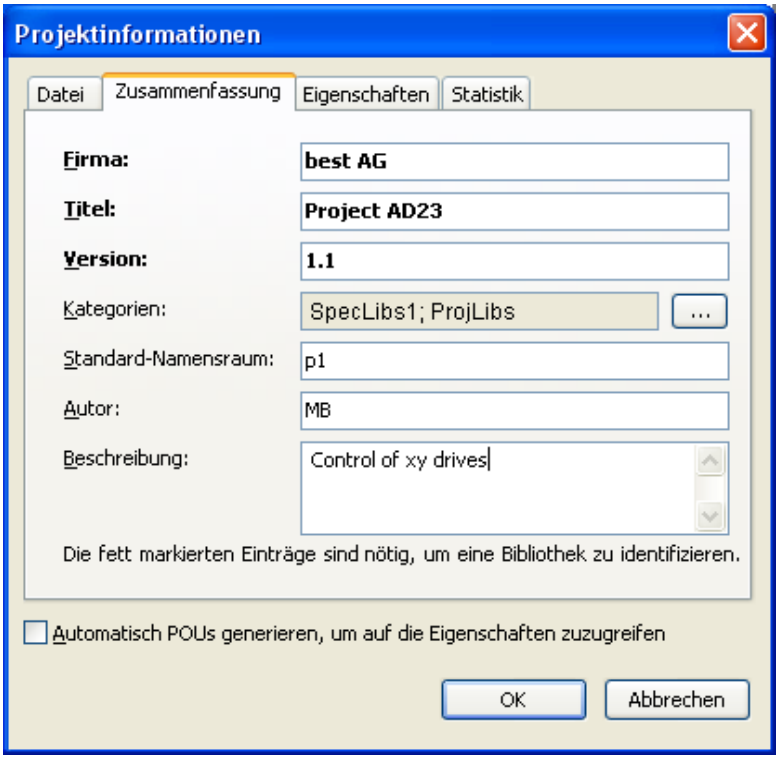


Abb. 149 Projektinformationen Dialog, Zusammenfassung

Hier kann folgende Information zur Projektdatei abgelegt werden: **Titel**, **Version**, ein **Standard-Namensraum**, der **Autor**, der Name der **Firma** und eine Kurz-**Beschreibung**. Diese Informationen werden automatisch als "Schlüssel" im Registerblatt 'Eigenschaften' verfügbar (s.u.).

Beachten Sie für Bibliotheksprojekte: Wenn eine Projekt als Bibliothek in anderen Projekten verwendet werden soll, muss hier zumindest ein **Titel**, eine **Version**snummer und der Firmenname hier eingetragen werden. Eine Bibliotheksdatei, die mit diesen Informationen versehen ist, kann auf dem System installiert und in Projekten eingebunden werden. Der **Firmenname** dient neben der Kategorie zur Sortierung im Bibliotheks-Repository-Dialog.

Um die Bibliothek gegen spätere Änderungen zu schützen, kann Option Freigegeben aktiviert werden. Dann erscheint - wenn das Projekt später erneut geöffnet wird - bei der ersten Aktion es zu editieren, eine Abfrage, ob es wirklich geändert werden soll. Man beachte, dass wenn diese mit Ja beantwortet wird, bei weiteren Editieraktionen keine weiteren Abfragen mehr erscheinen!

Optional können auch ein Standard-Namensraum, der Autor und eine Beschreibung mit angegeben werden, um als Bibliotheksprojekt-Information gespeichert zu werden. Wenn hier kein Standard-Namensraum definiert wird, gilt automatisch der Name der Bibliotheksdatei als Namensraum.

Zuweisung zu einer Kategorie:

Die Zuweisung der Bibliothek zu einer Kategorie dient ebenfalls der Sortierung im
Wird in den Projektinformationen keine Kategorie explizit angegeben, wird die Bibliothek der Kategorie "Sonstige" zugeordnet. Soll sie einer anderen Kategorie angehören, muss eine solche definiert sein. Die Definition von Bibliothekskategorien erfolgt in einer oder mehreren externen **Beschreibungsdateien** im XML-Format. Für die Zuordnung der Bibliothek kann entweder eine solche Datei hier aufgerufen werden, um die Kategorien verfügbar zu bekommen, oder aber eine andere Bibliotheksdatei, die bereits selbst die Informationen über die Kategorien aus einer Beschreibungsdatei aufgenommen hat.

Über Schaltfläche [...] wird der Dialog **Bibliotheks-Kategorien** geöffnet:

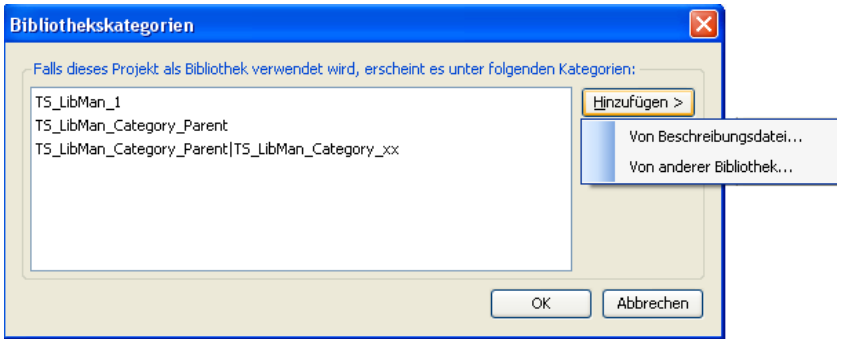


Abb. 150 Dialog zur Auswahl von Bibliothekskategorien

Schaltfläche **Hinzufügen** öffnet ein Menü, in dem gewählt werden kann, ob die Kategorien-Information **Aus einer Beschreibungsdatei...** oder **Aus einer anderen Bibliothek...** geholt werden soll. In beiden Fällen erscheint dann der Standarddialog zum Browsen nach einer Datei, wobei der Filter auf *.libcat.xml oder *.library gestellt ist.

Die Kategorien aus der Kategorien-Beschreibungsdatei oder der Bibliothek werden im Fenster aufgelistet. Nicht benötigte können Sie mit Schaltfläche **Entfernen** löschen. Weitere Kategorien aus anderen Quellen können auf die gleiche Weise hinzugefügt werden. Wenn die Liste alle gewünschten Kategorien enthält, bestätigen Sie mit **OK**, um den Dialog zu schließen und die Kategorien im Feld 'Bibliotheks-Kategorien im 'Projektinformationen'-Dialog einzutragen.

3. Eigenschaften

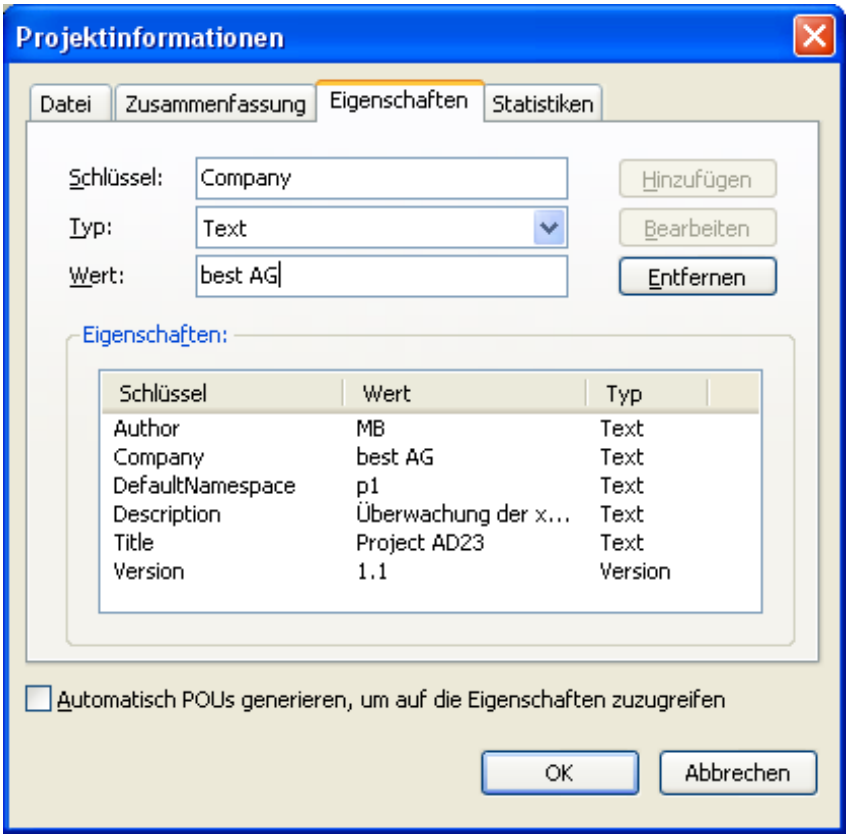


Abb. 151 Projektinformationen Dialog, Eigenschaften

Hier können Sie Schlüssel für bestimmte Datei-Eigenschaften definieren. Diese können dann in anwenderspezfiischen, externen Programmen verwendet werden, um die jeweilige Eigenschaften von außen zu kontrollieren.

Auf jeden Fall werden die Informationen, die im Registerblatt Zusammenfassung enthalten sind, als "Schlüssel" angeboten. Die Eigenschaften-Namen werden als Schlüsselnamen verwendet, als Datentyp wird automatisch "Text" eingestellt und die "Wert"e bestehen aus den Texten, die im Registerblatt 'Zusammenfassung' eingetragen sind. Weitere Schlüssel können jedoch explizit hinzugefügt werden (siehe im folgenden).

Hinzufügen eines Schlüssels:

Geben Sie im Feld **Schlüssel einen Namen ein und wählen Sie den gewünschten Datentyp von der Liste bei Typ:** (mögliche Datentypen: Text, Datum, Zahl, Wahr/Falsch, Version). Im Feld **Wert: geben Sie den gewünschten Wert ein, der zum Datentyp passen muss. Mit Schaltfläche Hinzufügen** können Sie nun den neuen Schlüssel in die **Eigenschaften**-Liste übertragen.

Schlüssel bearbeiten:

Wählen Sie in der Eigenschaften-Liste den Schlüssel durch Auswahl in der 'Schlüssel'-Spalte und ändern Sie die Schlüsselattribute wie gewünscht in den Eingabefeldern oberhalb der Liste. Daraufhin kann Schaltfläche **Bearbeiten** verwendet werden, um die Änderungen in die Eigenschaften-Liste und bei den betreffenden Schlüsseln **auch in das Registerblatt 'Zusammenfassung'(!)** zu übernehmen.

Schlüssel entfernen:

in der Eigenschaften-Liste den zu ändernden Schlüssel durch Auswahl in der 'Schlüssel'-Spalte und drücken Schaltfläche **Entfernen**.

Automatisch POUs generieren, um auf die Eigenschaften zuzugreifen: Wenn diese Option aktiviert ist, werden automatisch POU-Objekte vom Typ Funktion im POUs-Fenster angelegt, die verwendet werden können, um im Applikationsprogramm auf die Projekt-Eigenschaften zuzugreifen. Spezielle Funktionen werden in diesem Fall für die Eigenschaften 'Firma', 'Titel' und 'Version' erzeugt (GetCompany, GetTitle, GetVersion). Um auf explizit definierte Eigenschaften zugreifen zu können, steht für jeden Eigenschaften-Typ eine entsprechende Funktion bereit (GetTextProperty, GetBooleanProperty, GetNumberProperty, GetVersionProperty). In diesem Fall müssen Sie die entsprechende dieser Funktionen aufrufen und dabei den Eigenschaften-Schlüssel (wie im Registerblatt Eigenschaften definiert) als Input übergeben, damit der Eigenschaften-Wert zurückgeliefert wird.

Beispiel: Die folgende Eigenschaft wird im Registerblatt 'Eigenschaften' definiert: Schlüssel = nProp1, Typ = Nummer, Wert = 333. Um den Wert im Applikationsprogramm zu erhalten, rufen Sie die Funktion GetNumberProperty auf; z. B. showprop:=GetNumberProperty("nProp1"); showprop muss in diesem Fall als DINT deklariert sein.

4. Statistiken:

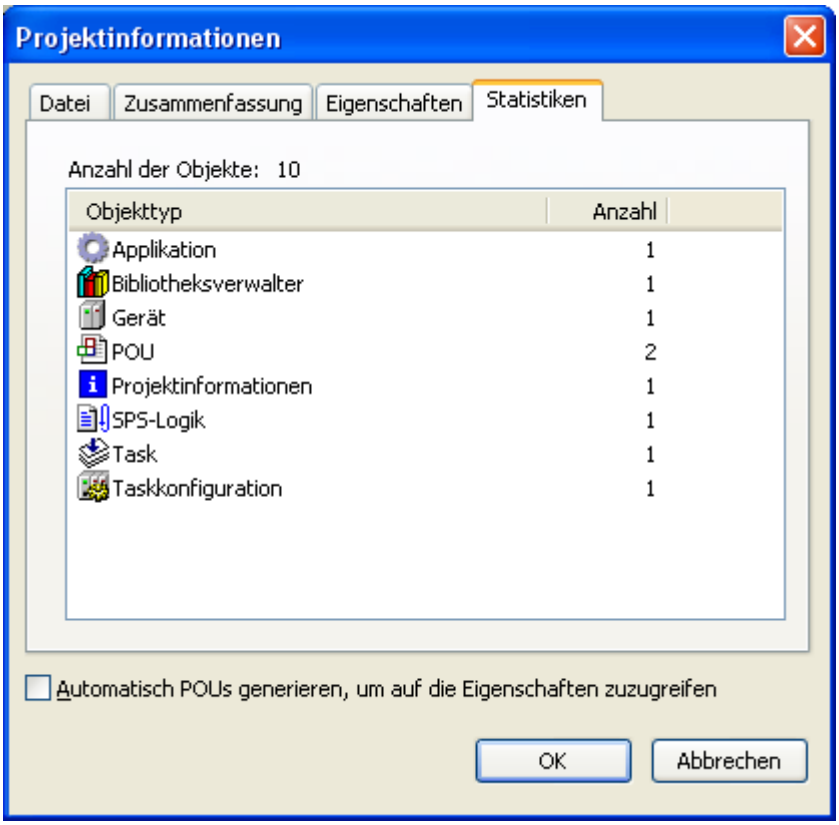


Abb. 152 Projektinformationen Dialog, Statistiken

Dieser Dialog zeigt, wieviel Objekte insgesamt (**Anzahl der Objekte**) und pro **Objekttyp** im Projekt verwendet werden (**Anzahl**).

Projekteinstellungen

Symbol:

Dieser Befehl (Kategorie 'Datei') ist standardmäßig in keinem Menü hinterlegt, aber in jedem neuen Projekt ist bereits automatisch das Objekt 'Projekteinstellungen' im POUs-Fenster eingefügt. Über einen Doppelklick auf dieses Objekt, was dem Ausführen des Befehls entspricht, wird der Dialog 'Projekteinstellungen' geöffnet:

Der Dialog enthält Unterdialoge für unterschiedliche Einstellungen wie z.B. Projektverschlüsselung, Benutzer- und Zugriffsrechte-Verwaltung, Versionshandling und Seiteneinstellungen für das Drucken.

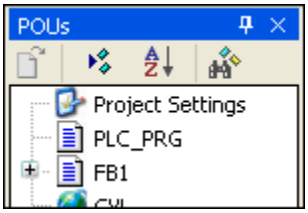


Abb. 153 Projekteinstellungen im POUs-Fenster

Im linken Teil des Dialogs sind die Kategorien der Einstellungen aufgelistet. Im rechten Teil wird zur gerade ausgewählten Kategorie ein Dialog geöffnet, in dem die Einstellungen bearbeitet werden können.

Kategorie Compile options (Übersetzungsoptionen)

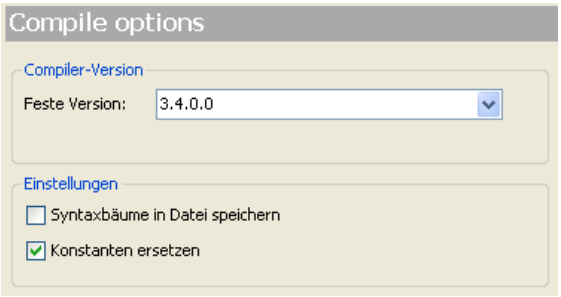


Abb. 154 Projekteinstellungen, Kategorie Compile options

Compiler-Version:

Hier ist festgelegt, welche Compiler-Version bei der Ausführung der Befehle 'Erstellen' oder während des Herunterladens zum Übersetzen des Projekts verwendet werden soll. Dazu kann im Feld **Feste Version** die gewünschte Version ausgewählt werden.

Hinweis: Ob der Datentyp LREAL als 64-Bit-Typ behandelt wird oder in REAL konvertiert wird, hängt vom Zielgerät ab. Für weitere Informationen sehen Sie bitte die dazugehörige Dokumentation.

Einstellungen:

Syntaxbäume in Datei speichern: Im Falle von großen Projekten oder weniger performanten Systemen kann es sinnvoll sein, diese Option zu aktivieren. Sie hilft Speicherprobleme zu vermeiden, indem die Syntaxbäume (Sprachmodell, Precompile Information) in einer temporären Datei anstatt im Hauptspeicher gehalten werden. Allerdings führt dies zu längeren Projektlade- und -übersetzungszeiten.

Konstanten ersetzen: Diese Option ist standardmäßig aktiviert, was bedeutet: Für jede Konstante skalaren Typs (also nicht für Strings, Arrays, Strukturen) wird direkt ihr Wert geladen. Im Online-Betrieb werden Konstanten im Deklarationseditor oder Überwachen-Fenster durch ein dem Wert vorangestelltes Symbol gekennzeichnet. In diesem Fall ist ein Zugriff, z.B. über einen ADR-Operator, Forcen und Schreiben, nicht möglich. Wenn die Option deaktiviert ist, kann auf die Konstante zugegriffen werden, allerdings verlängert dies die Rechenzeit.

Kategorie Seiteneinstellungen

Die Dialoge für das Konfigurieren des Layouts von Druckseiten für die Projektdokumentation werden vom 'Print Engine' Plug-in bereitgestellt. Sie sind auch über den Befehl 'Seiteneinstellungen' zugänglich.

Kategorie Quellcode-Download

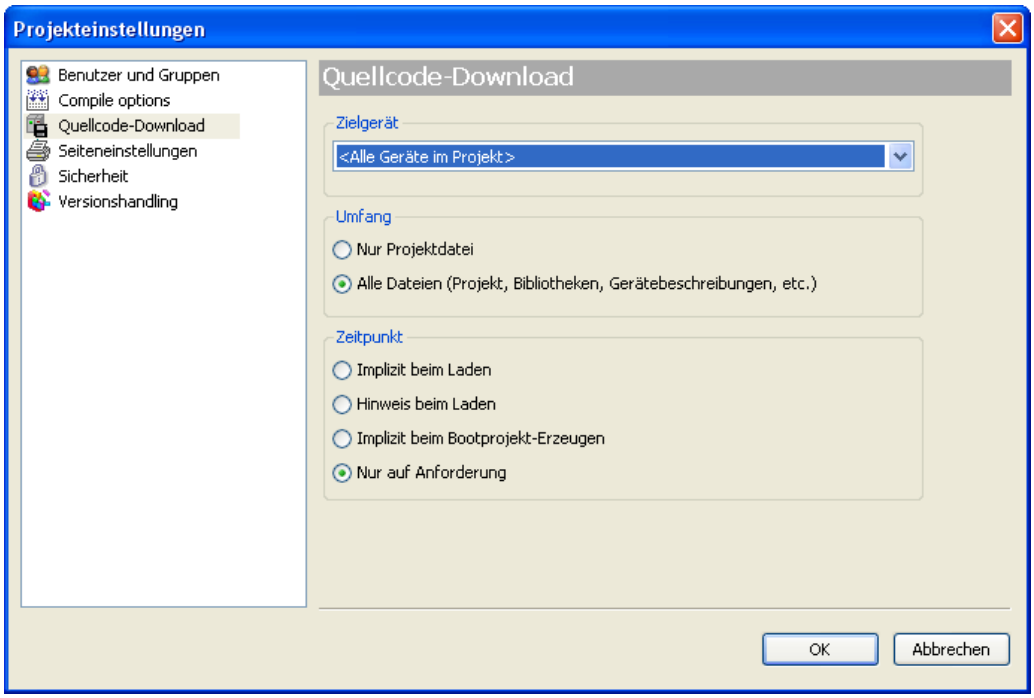


Abb. 155 Projekteinstellungen, Kategorie Quellcode-Download

Folgende Einstellungen sind zulässig für den Quellcode-Download, d.h.für die Übertragung und Speicherung der Projektdatei(en) auf die Steuerung.

Zielgerät:

Hier ist kein Gerät standardmäßig festgelegt. Sie können Ihre Auswahl aus einer Liste treffen, welche alle im Projekt definierten programmierbaren Geräte enthält. Nach Festlegung eines Zielgeräts kann kein Quellcode-Download vorgenommen werden, während Sie mit einem anderen Gerät verbunden sind.

Umfang:

Trifft eine Vorauswahl an zum Projekt gehörenden Dateien für eine Abspeicherung als Archivdatei.

Nur Projektdatei: nur die Datei <projectname>.project

Alle Dateien (Projekt, Bibliotheken, Gerätebeschreibungen, etc.): Standardeinstellung.

Zeitpunkt:

bestimmt, wann der Quellcode heruntergeladen wird:

Implizit beim Laden: Mit jedem Laden des ausführbaren Programms auf die Steuerung wird auch der Quellcode heruntergeladen (Befehl Online \ '<Application>' laden) ohne weitere Nachfrage beim Benutzer.

Hinweis beim Laden: Bei jedem Herunterladen des ausführbaren Programms auf die Steuerung wird der Benutzer gefragt, ob auch der Quellcode heruntergeladen werden soll.

Implizit beim Bootprojekt-Erzeugen: Der Quellcode wird bei jeder Erzeugung eines Boootprojekts (Befehl Online \ Bootapplikation erzeugen für '<Applikation>') ohne weiter Nachfrage beim Benutzer..

Nur auf Anforderung: Der Quellcode wird nur bei explizitem Aufruf des Befehls Online \ Quellcode auf Steuerung schreiben... heruntergeladen.

Kategorie Sicherheit

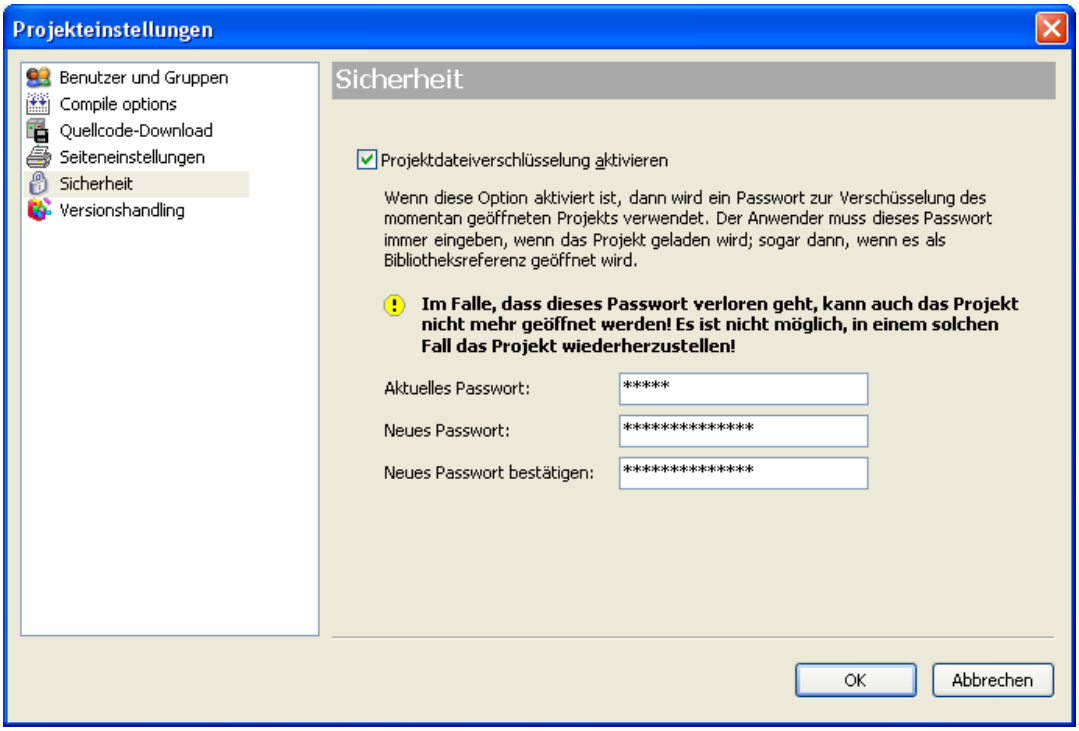


Abb. 156 Projekteinstellungen, Kategorie Sicherheit

Über diesen Dialog kann das Projekt mit einem Passwort geschützt werden. Nach Aktivierung der Option **Projektdateiverschlüsselung aktivieren**, kann das Passwort in den Feldern **Neues Passwort** und **Passwort bestätigen** (hier Passwort nochmals eingeben, um Tippfehler aufzudecken) eingegeben werden. Ist die Option bei Abspeicherung des Projekts weiterhin aktiviert, so muss das so definierte Passwort immer angegeben werden, wenn das Projekt wieder geladen wird, auch wenn es als Bibliotheksreferenz geöffnet

Dazu wird dann jeweils der Dialog 'Verschlüsselungspasswort' geöffnet.

Abb. 157 Dialog 'Verschlüsselungspasswort'

Achtung: Wenn das Verschlüsselungspasswort verloren wird, kann das Projekt nicht mehr geöffnet und auch nicht mehr restauriert werden!

Der Dialog in den Projekteinstellungen wird auch zur Abänderung eines bestehenden Passwortes verwendet. Dazu muss im Feld **Aktuelles Passwort** zunächst das alte Passwort eingegeben werden, bevor es in die neue Eingabe in den beiden Feldern **Neues Passwort** und **Passwort bestätigen** abgeändert werden kann.

Kategorie Benutzer und Gruppen

Die Dialoge für die Benutzer- und Rechteverwaltung werden mit dem Plug-In 'User Management UI' bereitgestellt. Siehe Projekteinstellungen, Benutzer und Gruppen.

Kategorie Visualisierung

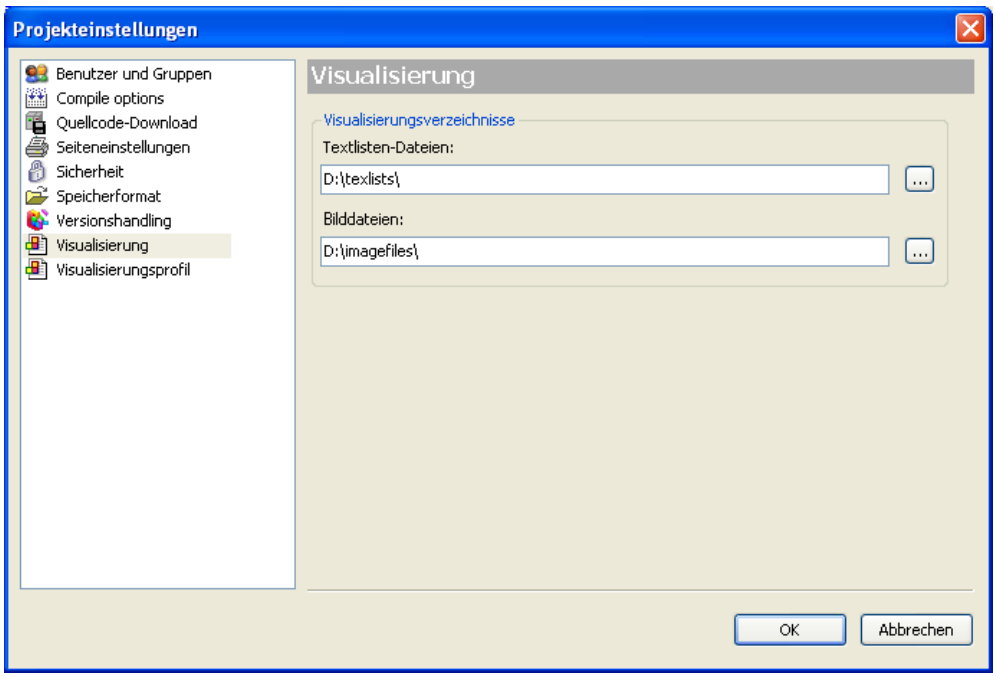


Abb. 158 Projekteinstellungen, Kategorie Visualisierung

Visualisierungsverzeichnisse: Die Standardeinstellungen für die folgenden Verzeichnisse sind durch die Projekt-Optionen vorgegeben.

Textlisten-Dateien: Öffnen Sie mit Schaltfläche ... den Standarddialog zur Definition eines existierenden oder eines neuen Verzeichnisses für Sprachdateien (Textlisten), die im Projekt zur Text- und Sprachenkonfiguration in der Visualisierung verwendet werden sollen. Die Dateien, die in diesem Verzeichnis gefunden werden, sind dann zu dem Zeitpunkt verfügbar, wenn eine Textliste importiert werden soll, z.B. nachdem sie von einem Übersetzer extern bearbeitet wurde. Es kann hier nur ein Verzeichnis angegeben werden.

Bilddateien: Öffnen Sie mit Schaltfläche ... den Standarddialog zur Definition eines existierenden oder eines neuen Verzeichnisses für die Bilddateien (Bildersammlungen), die im Projekt für die Verwendung in Visualisierungen bereitstehen sollen. Hier können mehrere Verzeichnisse in einer Strichpunkt-separierten Liste angegeben werden.

Kategorie Visualisierungsprofil

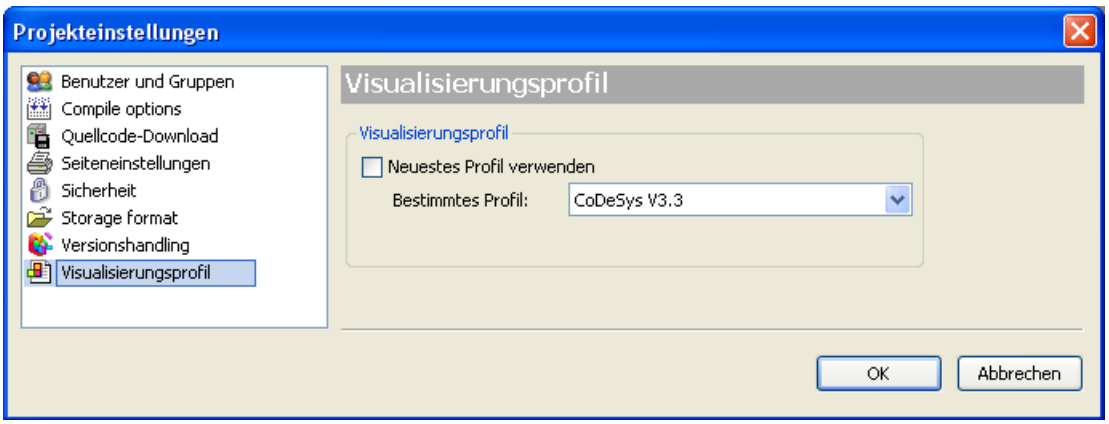


Abb. 159 Projekteinstellungen, Kategorie Visualisierungsprofil

Legen Sie hier fest, welches der aktuell auf dem System verfügbaren Visualisierungsprofile verwendet werden soll, wenn das Projekt geöffnet wird:

Bestimmtes Profil: Das aktuell aus der Auswahlliste eingestellte Profil wird verwendet.

Wenn ein Projekt geöffnet wird, das nicht das neueste verfügbare Visualisierungsprofil verwendet, wird darauf hingewiesen und der Anwender kann im Dialog Projektumgebung eine entsprechende Aktualisierung vornehmen.

Wenn die Einstellungen auch für weitere Projekte gelten sollen, die im aktuell geöffneten Programmiersystem erstellt werden, muss die Option **Diese Einstellung als Default für neue Projekte verwenden** aktiviert werden.

12.15.2 Projekteinstellungen - Benutzer und Gruppen

Der Dialog 'Projekteinstellungen' bietet in Kategorie 'Benutzer und Gruppen' drei Unterdialoge für die Benutzerverwaltung im aktuellen Projekt: Benutzer, Gruppen, Einstellungen.

Benutzer

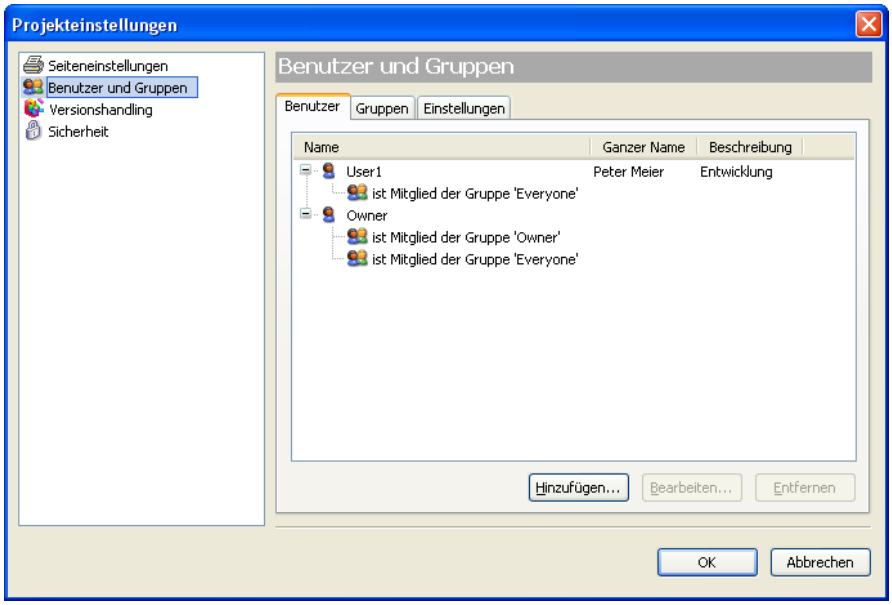


Abb. 160 Projekteinstellungen, Dialog 'Benutzer'

Alle bereits definierten Benutzer und ihre Mitgliedschaften sind hier in einer Baumstruktur dargestellt. Wenn ein Benutzerkonto über den Dialog 'Benutzer hinzufügen' oder 'Benutzer bearbeiten' (s.u.) angelegt wurde, erscheint neben dem Namen (Anmeldename) auch der ganze Name und eine Beschreibung. Die Gruppenmitgliedschaften eines Benutzers können mit Hilfe der Plus- und Minuszeichen vor dem Namen angezeigt bzw. ausgeblendet werden. Jeder Benutzer ist per Default Mitglied der Gruppe "Everyone".

Neues Benutzerkonto anlegen:

Öffnen Sie über Schaltfläche Hinzufügen den Dialog Benutzer hinzufügen:

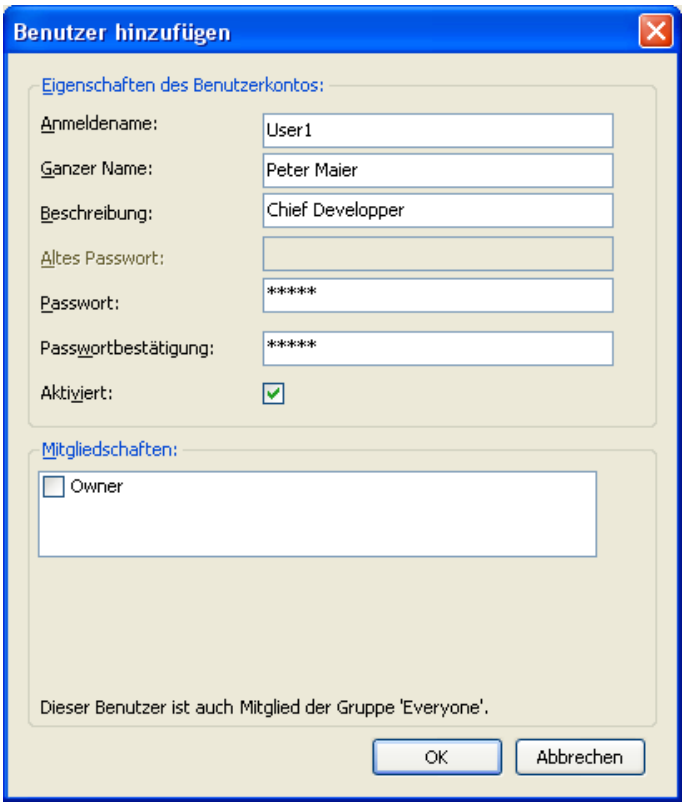


Abb. 161 Dialog 'Benutzer hinzufügen'

Füllen Sie die folgenden Eingabefelder:

Eigenschaften des Benutzerkontos:

- Anmeldename: Name, mit dem sich der Benutzer anmelden kann.
- Ganzer Name: Voller Name des Benutzers, nur zur zusätzlichen Information.
- Beschreibung: Beschreibung des Benutzers, nur zur zusätzlichen Information.
- Altes Passwort: Dieses Feld ist nur editierbar, wenn der Dialog zum Ändern des Benutzerkontos verwendet wird ("Benutzer bearbeiten", siehe unten). Um das Passwort eines bestehenden Benutzers ändern zu können, muss erst das aktuell gültige Passwort eingegeben werden.
- Passwort: Passwort für den Benutzer. Die Eingabe ist mit *-Zeichen maskiert.
- Passwortbestätigung: Die bei 'Passwort' eingegebene Zeichenfolge muss hier wiederholt werden. Wenn die Eingaben nicht übereinstimmen, wird eine Fehlermeldung ausgegeben. Die Eingabe ist ebenfalls mit *-Zeichen maskiert.
- Aktiviert: Wenn diese Option aktiviert ist (Default), ist das Benutzerkonto ebenfalls aktiviert und kann verwendet werden. Wenn die Option nicht aktiviert ist, kann der Benutzer sich nicht anmelden. Eine Deaktivierung des Kontos kann automatisch erfolgen, wenn wiederholt ein Einloggen mit unkorrekten Anmeldedaten versucht wurde.

- Mitgliedschaften: Hier werden alle Benutzergruppen aufgelistet, die zusätzlich zur Gruppe "Everyone" (zu der jeder neue Benutzer automatisch gehört) definiert wurden. Durch Aktivieren der jeweiligen Gruppeneinträge (☑) wird festgelegt, zu welchen Gruppen der neue Benutzer gehört.

Bestätigen Sie das neue Benutzerkonto mit OK. Auf unkorrekte oder fehlende Eingaben (kein Anmeldename, Benutzer ist bereits registriert etc.) wird durch entsprechende Fehlermeldungen aufmerksam gemacht.

Bestehendes Benutzerkonto bearbeiten:

Selektieren Sie die Zeile mit dem zu ändernden Benutzereintrag und öffnen über Schaltfläche Bearbeiten den Dialog Benutzer bearbeiten. Dort finden Sie dieselben Eingabefelder wie im Dialog 'Benutzer hinzufügen' (s.o.). Nur die Passwortfelder sind hier - aus Sicherheitsgründen - immer mit 32 *-Zeichen gefüllt. Sie können die Einträge wie gewünscht bearbeiten und durch Bestätigen mit OK wirksam machen.

Eines oder mehrere Benutzerkonten löschen:

Selektieren Sie die entsprechenden Benutzer in der Liste und löschen Sie die Konten mit Schaltfläche Entfernen. Beachten Sie, dass keine weitere Nachfrage erscheinen wird! Eine entsprechende Fehlermeldung wird ausgegeben, wenn Sie versuchen, den letzten Benutzer einer Gruppe zu löschen. Eine Gruppe muss mindestens ein Mitglied haben.

Gruppen

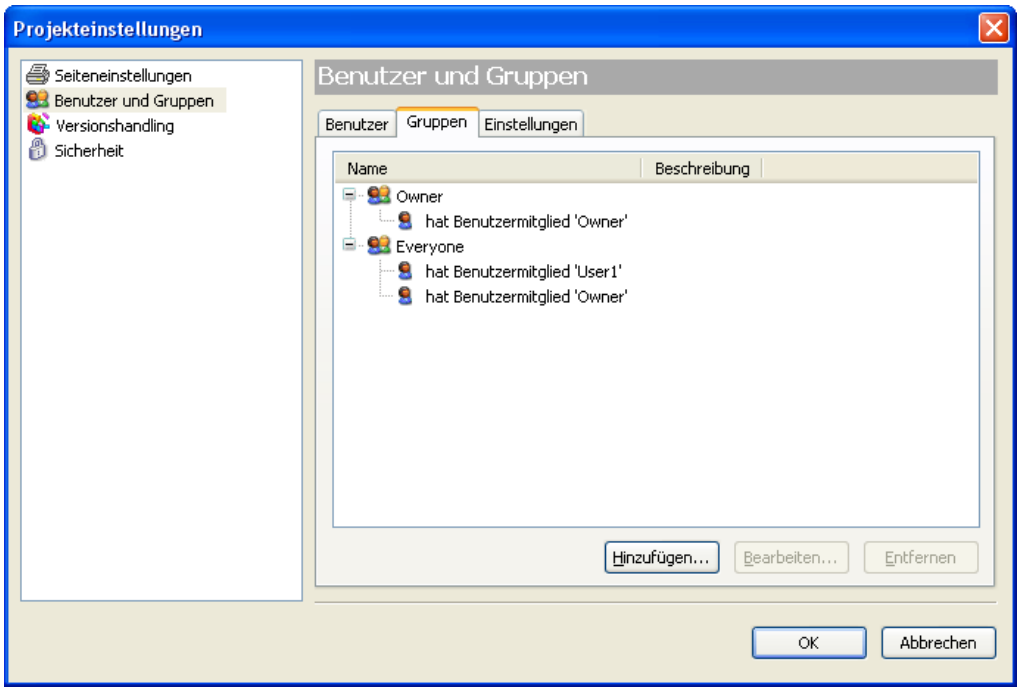


Abb. 162 Projekteinstellungen, Dialog 'Gruppen'

Alle definierten Gruppen und die zugehörigen Benutzer werden in einer Baumstruktur dargestellt. Die Mitglieder jeder Gruppe können mit Hilfe der Plus- und Minuszeichen vor dem Gruppennamen angezeigt bzw. ausgeblendet werden. Ein Mitglied kann wiederum eine Gruppe sein.

Neue Gruppe hinzufügen:

Öffnen Sie über Schaltfläche Hinzufügen den Dialog Gruppe hinzufügen:

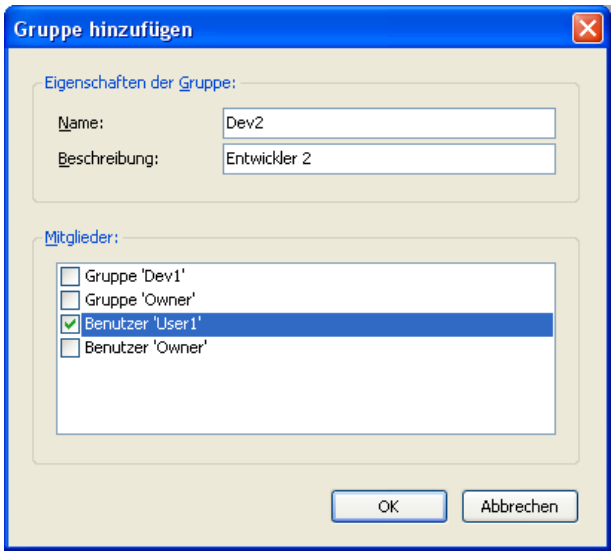


Abb. 163 Dialog 'Gruppe hinzufügen'

Füllen Sie die folgenden Eingabefelder:

Eigenschaften der Gruppe:

- Name: Gruppenname.
- Beschreibung: Beschreibung, nur zur zusätzlichen Information.
- Mitglieder: Liste aller existierenden Benutzer und Gruppen. Wählen Sie diejenigen aus (☑), die Mitglieder der neuen Gruppe sein sollen.

Bestätigen Sie die neue Gruppe mit OK. Auf unkorrekte oder fehlende Eingaben (kein Name angegeben, Gruppe existiert bereits, als Mitglied wurde eine Gruppe ausgewählt, die eine "Grupenrückkopplung" verursachen würde) wird durch entsprechende Fehlermeldungen aufmerksam gemacht.

Bestehende Gruppe bearbeiten:

Wählen Sie die Gruppe in der Liste aus und öffnen über Schaltfläche Bearbeiten den Dialog Gruppe bearbeiten. Dort finden Sie dieselben Eingabefelder wie im Dialog 'Gruppe hinzufügen' (s.o.). Nach Bearbeiten der Eingaben bestätigen Sie mit OK um die Änderungen wirksam zu machen.

Eine oder mehrere Gruppen löschen:

Wählen Sie die entsprechenden Einträge in der Gruppenliste aus und entfernen Sie sie mit Entfernen. Beachten Sie, dass keine weitere Nachfrage erscheinen wird! Die Konten der Mitglieder der gelöschten Gruppen bleiben unverändert. Eine entsprechende Fehlermeldung wird ausgegeben, wenn versucht wird, die Gruppen "Everyone" und/oder "Owner" zu löschen.

Einstellungen

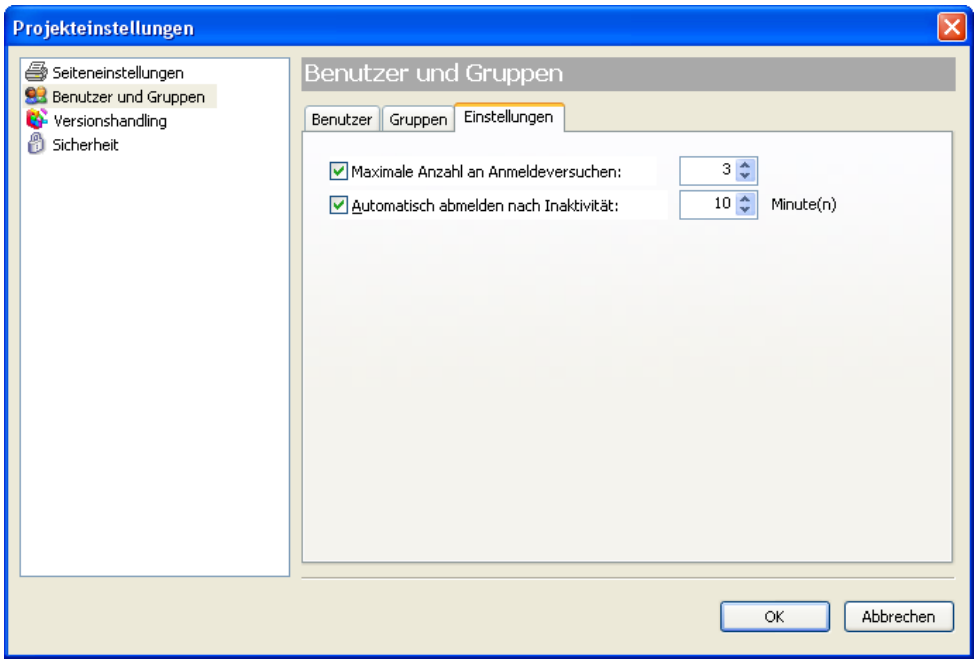


Abb. 164 Projekteinstellungen, Dialog 'Einstellungen'

Hier können folgende allgemeine Einstellungen für alle Benutzerkonten vorgenommen werden:

- Maximale Anzahl an Anmeldeversuchen: Wenn diese Option aktiviert ist, wird ein Benutzerkonto deaktiviert, sobald der Benutzer so oft, wie hier angegeben, versucht hat , sich mit einem falschen Passwort anzumelden. Wenn die Option nicht aktiviert ist, ist die Zahl der Fehlversuche unbegrenzt, d.h. es kommt nicht zur Deaktivierung des Kontos. Default: Option aktiviert, Anzahl Fehlversuche:3; Mögliche Eingabewerte: 1-10.
- Automatisch Abmelden nach Inaktivität: Wenn diese Option aktiviert ist, wird der Benutzer automatisch abgemeldet, sobald über die hier angegebenen Zeitspanne (Minuten) keine Benutzeraktionen im Programmiersystem über Maus oder Tastatur mehr registriert wurden. Default: Option aktiviert, Zeit: 10 Minuten; Mögliche Angaben: 1-180)

12.16 Hilfe

Die Befehlskategorie 'Hilfe' enthält Befehle zum Handhaben der Online Hilfe sowie für das Erhalten weiterer Informationen zu »PLC Designer«.

12.16.1 Befehle der Kategorie Hilfe

Das Plug-In 'Online Help' stellt Befehle der Kategorie 'Hilfe' bereit, mit denen die Online Hilfe des Entwicklungssytems bedient werden kann.

Standardmäßig sind die Befehle im Menü 'Hilfe' zu finden. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

Die Befehle:

- Inhalt...
- Index...
- Suchen...
- Informationen...
- Synchronisieren

Weitere Befehle können verfügbar sein, die eine Verknüpfung zu einer Internet-Seite herstellen.

12.16.2 Inhalt...

Standard Tastenkürzel: <Umsch>+<Strg>+<F1>

Dieser Befehl der Kategorie Hilfe öffnet das Inhaltsverzeichnis der Online Hilfe in einem separaten Tabulator des Hilfefensters.

Das Inhaltsverzeichnis ist mit Hilfe von "Büchern" strukturiert, die mit einem Mausklick auf das vorangestellte Plus- bzw Minuszeichen geöffnet bzw. geschlossen werden können. Eine Buchseite wird nach einem Mausklick auf den Seiteneintrag im Inhaltsverzeichnis in einem eigenen Fenster angezeigt.

12.16.3 Index...

Standard Tastenkürzel: <Umsch>+<Strg>+<F21>

Dieser Befehl der Kategorie Hilfe öffnet in einem separaten Tabulator des Hilfefensters einen Dialog, der alle Indexeinträge der Online Hilfe auflistet.

Wenn im Feld **Suchen nach:** eine Zeichenfolge eingegeben wird, wird der erste Indexeintrag, der dem Suchbegriff entspricht, in der Liste selektiert. Durch einen Doppelklick auf diesen Eintrag wird entweder die zugehörige Hilfeseite geöffnet, oder - wenn dem Eintrag mehrere Seiten zugewiesen sind - eine Liste dieser Seiten in einem separaten Fenster "Indexergebnisse" angezeigt.

12.16.4 Suchen...

Dieser Befehl der Kategorie Hilfe öffnet in einem separaten Tabulator des Hilfefensters den Suchen-Dialog für das Suchen nach einem bestimmten Begriff bzw. einer Zeichenfolge in der Online Hilfe. Dabei wird eine Volltextsuche über alle Hilfeseiten durchgeführt. Die zu suchende Zeichenfolge wird im Feld **Suchen nach:** eingegeben.

Die folgenden Suchoptionen können aktiviert werden:

Nur in Titeln suchen: Nur die Titelzeilen der Hilfeseiten werden durchsucht.

Teiltreffer anzeigen: Die Zeichenfolge wird auch gefunden, wenn sie Teil einer anderen Zeichenfolge ist. Beispiel: "log" wird auch in "dialog" gefunden. Wenn die Option nicht aktiviert ist, wird die Zeichenfolge nur gefunden, wenn sie als separat steht: "log".

Auf ... Treffer beschränken: Die Anzahl der gefundenen Stellen, die im Suchergebnisse-Fenster angezeigt werden sollen, kann hier begrenzt werden.

Die Liste der Hilfeseiten, die die gesuchte Zeichenfolge enthalten, werden gemäß der obigen Optionen in einem eigenen Fenster "Suchergebnisse" aufgelistet, das zu diesem Zweck automatisch geöffnet wird.

12.16.5 Informationen...

Dieser Befehl der Kategorie Hilfe öffnet eine Box, die Informationen (Versionsnummer etc.) zum »PLC Designer« Entwicklungssystem enthält.

12.16.6 Synchronisieren

Dieser Befehl der Kategorie Hilfe aktualisiert das Inhaltsverzeichnis der Online Hilfe in der Hinsicht, dass die gerade geöffnete Hilfeseite im Inhaltsverzeichnis selektiert wird. Wenn nötig, wird das entsprechende Buch dazu aufgeklappt.

12.17 Bildersammlung

Die Befehlskategorie 'Bildersammlung' enthält Befehle zum Erstellen und Konfigurieren von Bildersammlungen.

12.17.1 Befehle zu Bildersammlung

Die Befehle zum Arbeiten mit Bildersammlungen werden vom Plug-In 'Image pool' bereitgestellt.

Gegebenenfalls kann die Menüstruktur über den Anpassen-Dialog umkonfiguriert werden.

- Die Befehle:
- Bildersammlung erzeugen

12.17.2 Globale Bildersammlung erzeugen

Mit diesem Befehl der Kategorie Bildersammlung wird die globale Bildersammlung "GlobalImagePool" manuell angelegt. GlobalImagePool ist eine Sammlung für statische Bilder, die in Visualisierungen verwendet werden. Beachten Sie, dass GlobalImagePool automatisch erzeugt wird, sobald die erste statische Bilddatei-Verwendung in einem Projekt definiert wird.

12.17.3 Bildersammlung erzeugen

noch nicht verfügbar

12.18 Installation

Diese Befehlskategorie enthält Befehle zum Installieren/Deinstallieren von Bibliotheken und Visualisierungselementen.

12.18.1 Bibliotheks-Repository

Befehle der Kategorie Installation - Bibliotheks-Repository

Das Plug-In 'Library Manager Object' stellt für die Kategorie 'Installation' den Befehl 'Bibliotheks-Repository...' zum Öffnen des gleichnamigen Dialogs bereit.

Standardmäßig ist der Befehl im 'Tools'-Menü zu finden. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog verändert werden.

Allgemeine Informationen zur Verwendung von Bibliotheken: Siehe Bibliotheksverwaltung**.**

Bibliotheks-Repository...

Symbol:

Hinweis: Es hängt vom aktuell verwendeten Feature-Set ab, ob diese Funktionalität verfügbar ist !

Dieser Befehl (Kategorie Installation) öffnet den **Bibliotheks-Repository** Dialog, der auch aus dem Bibliotheksverwalter-Editor geöffnet werden kann. Ein Bibliotheks-Repository ist ein Ablageort für Bibliotheken, die auf dem lokalen System installiert wurden, um in »PLC Designer« Projekte eingebunden werden zu können.

Hinweis: Ein Bibliotheksprojekt *.library, das in einem Bibliotheks-Repository liegt, kann nicht aus diesem heraus zur Bearbeitung bzw. Ansicht im Programmiersystem geöffnet werden.

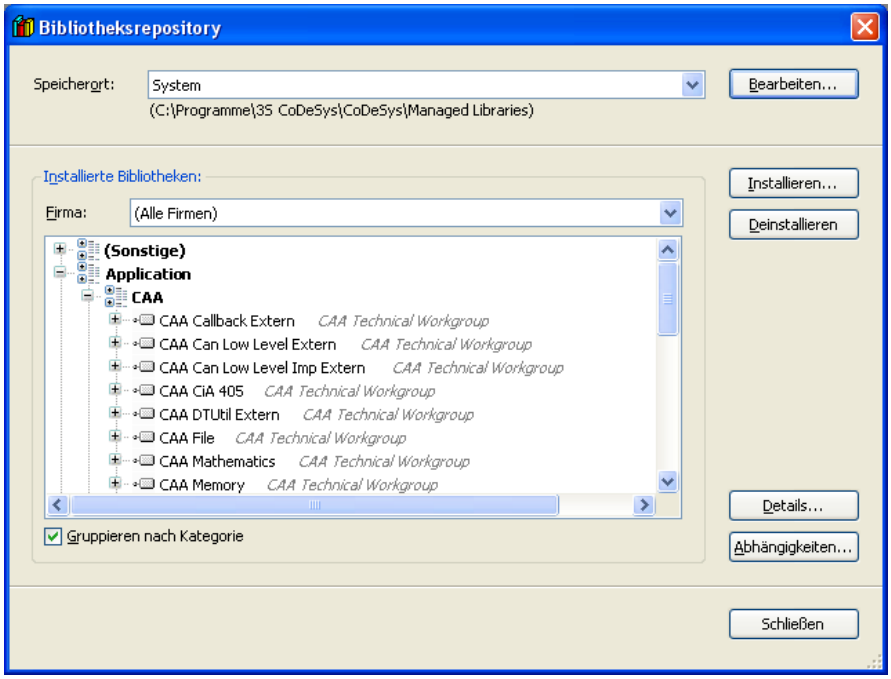


Abb. 165 Bibliotheks-Repository Dialog

Allgemeine Informationen zur Verwendung von Bibliotheken: Siehe Bibliotheksverwaltung.

Der Dialog zeigt die momentan definierten Speicherorte (Repositories) und die dort installierten Bibliotheken. Repositories können über diesen Dialog neu hinzugefügt, modifiziert oder entfernt werden und Bibliotheken können hier installiert oder deinstalliert werden.

Für den gerade ausgewählte **Speicherort**, ein Verzeichnis auf dem lokalen System, in dem die Bibliotheksdateien liegen, und die gerade eingestellte **Firma** werden alle aktuell **Installierten Bibliotheken** in einer Liste angezeigt. Diese Liste zeigt jeweils Namen (Titel), Version und Firma einer Bibliothek, wie sie in den Projektinformationen der Bibliotheksdatei angegeben sind.

Wenn die Option **Gruppieren nach Kategorie** aktiviert ist, wird die Liste nach Bibliothekskategorien gruppiert, wobei die Kategorienamen als Knoten erscheinen, die durch einen Mausklick aufgeklappt oder zugeklappt werden können. Unter den Kategorien erscheinen die jeweiligen Bibliotheken und wiederum darunter die jeweils verfügbaren Versionen. Wenn nicht nach Kategorien gruppiert ist, werden die Bibliotheken pro Firma alphabetisch gelistet. Die Kategorien werden durch externe Beschreibungsdateien "*.libcat.xml" definiert. Siehe Bibliotheksverwaltung für weitere Informationen hierzu.

Zu den einzelnen Schaltflächen finden Sie hier auf der Seite die folgenden Beschreibungen:

- **Bearbeiten**: Bibliotheks-Repositories
- **Installieren**, **Deinstallieren**: Bibliotheken installieren und deinstallieren
- **Details**, **Abhängigkeiten**: Weitere Informationen zu einer Bibliothek

Bibliotheks-Repositories

Ein oder mehrere Repositories können für die Verwaltung von Bibliotheken verwendet werden. Alle gerade definierten Repositories stehen in der Auswahlliste hinter **Speicherort**: Standardmäßig wird auf jeden Fall das "System" Repository durch die »PLC Designer« Installation angelegt..

Um Pfad oder Namen eines Repositories zu ändern verwenden Sie die Schaltfläche **Bearbeiten...** um den Dialog **Repositories bearbeiten** zu öffnen:

Abb. 166 Dialog 'Repository-Speicherorte bearbeiten'

Die aktuell definierten Speicherorte werden im Repositories-Fenster aufgelistet. Sie werden in der hier vorliegenden Reihenfolge später nach einer Bibliothek durchsucht. Um die Reihenfolge zu verändern, können die Schaltflächen **Nach oben** und **Nach unten** verwendet werden, um den gerade ausgewählten Eintrag zu verschieben.

Beachten Sie, dass Speicherort "**<Alle Firmen>**" die Bibliotheken aller augenblicklich definierten Repositories auflistet. In dieser Darstellung ist keine Installation oder Deinstallation möglich.

Definieren eines neuen Repositories bzw. Bearbeiten des Pfades oder Namens eines bestehenden Repositories

Um ein neues Repository anzulegen, verwenden Sie Schaltfläche **Hinzufügen**. Der Dialog **Speicherort für Repositories** öffnet, wo Sie im Feld **Ort** den Pfad des neuen Repositories eingeben müssen. Sie können dabei über Schaltfläche [...] nach einem geeigneten Verzeichnis suchen bzw. eines anlegen. Achtung: Das ausgewählte Verzeichnis muss leer sein! Im Feld **Name** geben Sie einen symbolischen Namen für das Repository an, z.B. "Bibliotheken für System xy.

Um ein bestehendes Repository zu bearbeiten, wählen Sie den Eintrag im Dialog 'Repositories bearbeiten' aus und öffnen über Schaltfläche **Bearbeiten** ebenfalls den Dialog **Speicherort für Repositories**, wo in diesem Fall die aktuellen Eingaben für Ort und Name bereits eingetragen sind und verändert werden können.

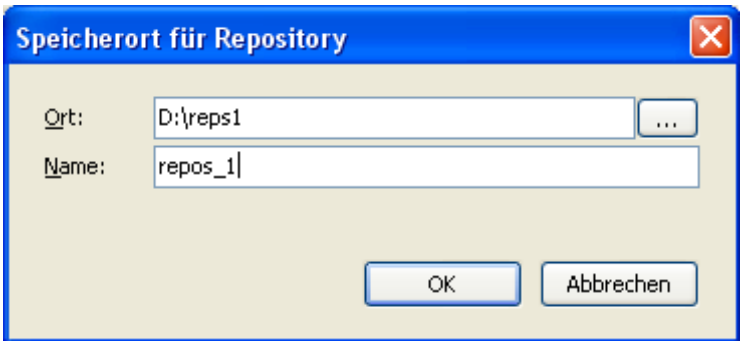


Abb. 167 Dialog 'Speicherort für Repositories', Repository hinzufügen oder bearbeiten

Hinweis: 1. Nur leere Verzeichnisse können für ein neues Repository verwendet werden. 2. Das "System"-Repository ist nicht bearbeitbar; dies wird durch die Schrägschrift des Eintrags angezeigt.

Löschen eines Repositories:

Wenn ein Repository in der Liste ausgewählt und die Schaltfläche **Entfernen** gewählt wird, erscheint eine Abfrage, ob nur der Eintrag aus der Liste der Repositories entfernt werden soll, oder ob auch das Verzeichnis mit den Bibliotheksdateien aus dem Dateisystem gelöscht werden soll.

Bibliotheken installieren und deinstallieren

Nur eine Bibliothek, die auf dem lokalen System in einem Bibliotheks-Repository installiert ist, kann über den Bibliotheksverwalter in ein Projekt eingebunden werden. Als Voraussetzung für die Installation muss die Bibliotheksdatei in ihren Projektinformationen einen Titel und eine Versionsinfo enthalten, optional ist die Angabe einer Kategorie und der Firma

Um eine Bibliothek zu installieren, wählen Sie im Bibliotheks-Repository Dialog das Repository aus, dem die Bibliothek hinzugefügt werden soll und aktivieren Schaltfläche **Installieren....**

Der Dialog 'Bibliothek auswählen' wird geöffnet, in dem nach der Bibliotheksdatei gesucht werden kann. Standardmäßig ist der Filter auf '»PLC Designer« Bibliotheken" gesetzt , die die entsprechenden Projektinformationen mitbringen und per Default die Erweiterung "*.library" tragen. Der Filter kann auch auf "»PLC Designer« Bibliotheken (vor 3.0)" gesetzt werden, um Bibliotheken alten Formats und mit Erweiterung "*.lib" zu finden, oder auf 'Alle Dateien'.

Wählen Sie die gewünschte Bibliotheksdatei aus und schließen den Dialog. Die Bibliothek wird zur Liste installierter Bibliotheken im Bibliotheks-Repository-Dialog hinzugefügt.

Wenn sie versuchen eine Bibliothek zu installieren, die nicht installierbar ist, weil sie nicht die nötigen Projektinformationen (Titel, Version) enthält, erscheint eine entsprechende Fehlermeldung.

Um eine Bibliothek zu deinstallieren, wählen Sie sie in der Liste installierter Bibliotheken im Bibliotheks-Repository-Dialog aus und aktivieren Schaltfläche **Deinstallieren**.

Weitere Informationen zu einer Bibliothek

Details...: Wenn in der Liste installierter Bibliotheken eine Bibliothek ausgewählt wird - es muss die Zeile selektiert werden, die die Version angibt! - , öffnet diese Schaltfläche einen Dialog, in dem folgende Detailinformationen ausgegeben werden: Titel, Version, Firma, Größe, Erstellt (Datum), Geändert (Datum), Bearbeitet (Datum letzter Zugriff), Eigenschaften. Die Angaben stammen aus den Projektinformationen der Bibliotheksdatei.

Abb. 168 Dialog 'Details'

Abhängigkeiten... Für die gerade ausgewählte Bibliothek öffnet diese Schaltfläche einen Dialog, der die Abhängigkeiten von anderen Bibliotheken darstellt. Das heißt, es werden die Bibliotheken dargestellt, die in ihr eingebunden sind. Für jede Bibliotheksreferenz werden Titel, Version und Firma angezeigt.

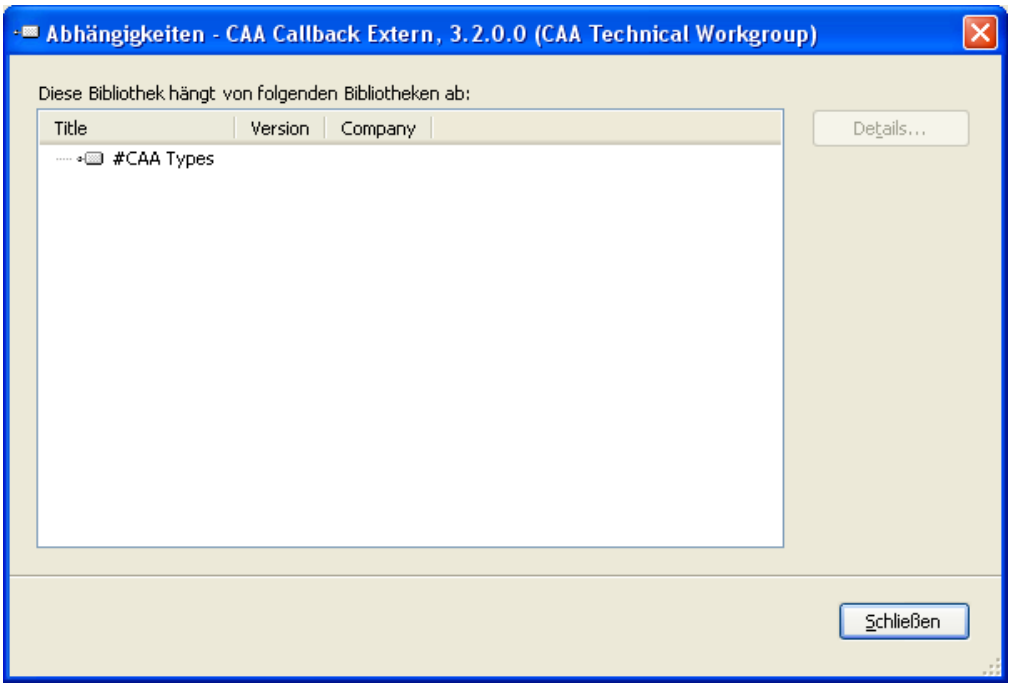


Abb. 169 Dialog 'Abhängigkeiten'

12.18.2 Visualisierungselemente-Repository

Befehle der Kategorie Installation - Visualisierungselemente-Repository

Das Plug-In 'VisualElement Repository' stellt für Befehlskategorie 'Installation' den Befehl Visualisierungselemente-Repository bereit, mit dem auf dieses Repository zugegriffen werden kann.

Der Befehl ist standardmäßig im Visualisierung-Menü zu finden. Wenn erforderlich kann die Menüstruktur mit Hilfe des Anpassen-Dialogs verändert werden (Kategorie "Installation").

Beachten Sie, dass der Befehl **nur verfügbar ist, wenn kein Projekt geöffnet ist**. Dies beruht darauf, dass ein definiertes Set der Visualisierungselemente-Repository-Einstellungen als "Visualisierungsprofil" verwaltet wird und die verschiedenen Visualisierungsprofile unabhängig von einem Projekt zur Verfügung stehen.

Visualiserungselemente-Repository

Dieser Befehl (Kategorie 'Installation') öffnet den Dialog **Repository für Visualierungselemente**, wo Element-Repositories und Visualisierungsprofile definiert werden können.

Beachten Sie, dass sobald der Dialog über Schaltfläche **Schließen** verlassen wird, die aktuellen Einstellungen unter dem Profilnamen, der gerade im Profilauswahl-Feld angezeigt wird.

Ein Visualisierungsprofil definiert, welche Visualisierungselemente (bereitgestellt durch die verwendeten Elementbibilotheksversionen) in der WerkzeugBox enthalten sind, wenn im Visualisierungs-Editor in einem Projekt gearbeitet wird, das auf diesem Profil basiert. Jedes Visualisierungsprojekt muss ein Visualisierungsprofil verwenden und es ist in den Projekteinstellungen definiert, welches standardmäßig verwendet wird.

Ein Profil kann nur außerhalb eines Projekts konfiguriert werden. Deshalb ist der "Visualisierungslemente-Repository" Editor **nur verfügbar, wenn kein Projekt geöffnet ist**.

Im oberen Teil des Visualisierungselemente-Repository-Dialogs können die Ablageorte der Repositories bearbeitet werden. Sehen Sie dazu unten eine Beschreibung des Dialogs Repositories bearbeiten. Im unteren Teil können die Visualisierungsprofile bearbeitet werden. Sehen Sie dazu unten die Beschreibung Visualisierungsprofile bearbeiten.

Repositories bearbeiten

Dieser Teil des Visualisierungselemente-Repository-Dialogs enthält eine Auswahlliste der aktuell konfigurierten Speicherorte. Um einen Speicherort zu bearbeiten oder hinzuzufügen, bedienen Sie die Schaltfläche **Speicherorte bearbeiten...**.

Im unteren Teil des Dialogs befinden sich auf zwei Registerblättern jeweils für den gerade eingestellten Speicherort die Liste der bereits **Installierten Elemente** und aller **Verfügbaren Elemente**.

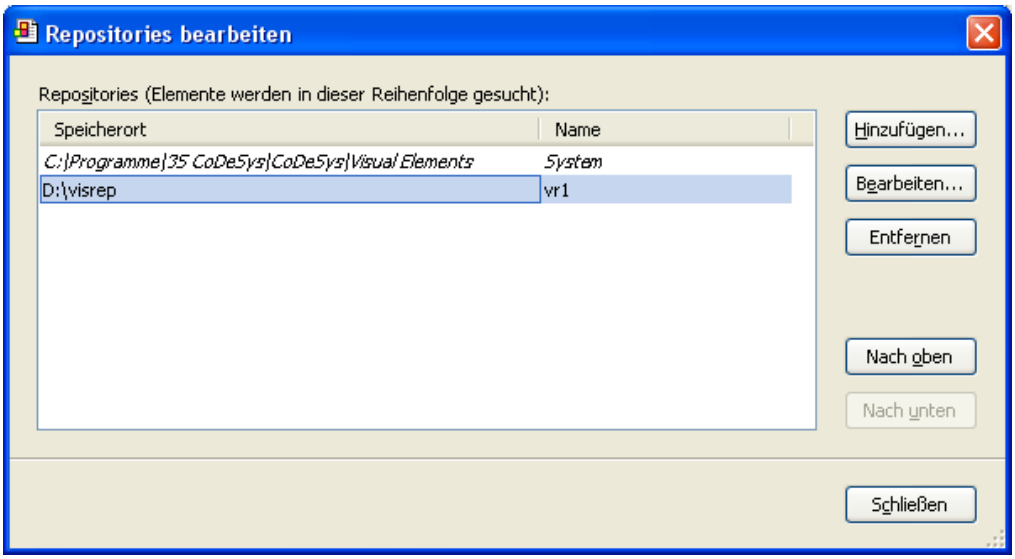


Abb. 170 Dialog 'Repositories bearbeiten'

Für die Verwaltung der Visualisierungselemente können ein oder mehrere Repositories verwendet werden. Alle aktuell definierten Speicherorte sind hier aufgelistet. Die Reihenfolge von oben nach unten ist dann auch die Suchreihenfolge für Visualisierungselemente. Automatisch ist immer ein Speicherort "System" definiert, der auch nicht bearbeitet oder gelöscht werden kann.

Bearbeiten oder Hinzufügen eines Repositories:

Um Pfad oder Namen eines Repositories zu ändern, wählen Sie dieses in der Liste aus und öffnen über Schaltfläche **Bearbeiten** den Dialog **Speicherort für Repository**.

Der gleiche Dialog wird verwendet um ein neues Repository zu definieren. Dazu wird er über Schaltfläche **Hinzufügen...** geöffnet.

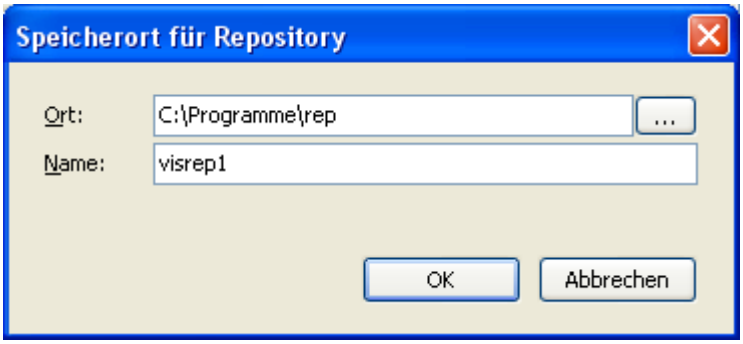


Abb. 171 Dialog 'Speicherort für Repository'

Die folgenden Eingaben sind zu machen bzw. zu verändern:

Im Feld **Ort** wird der Pfad des neuen Speicherorts eingegeben. Dazu kann über die Schaltfläche ... ein Dialog zum Suchen nach einem Ordner auf dem lokalen System verwendet werden. Achtung: Der gewählte Ordner muss leer sein! Im Feld **Name** geben Sie einen symbolischen Namen für das Repository ein, z.B. "Elemente Kategorie 1".

Hinweis: 1. Nur ein leeres Verzeichnis kann als neuer Speicherort für ein Repository gewählt werden. 2. Das "System" Repository kann nicht bearbeitet werden. Dies wird durch die Schrägschrift des Eintrags in der Repository-Liste angezeigt.

Repository löschen:

Mit **Entfernen** wird das gerade n der Repository-Liste selektierte Repository gelöscht.

Visualisierungsprofile bearbeiten

In diesem Teil des Visualierungselemente-Repository- Dialogs können Visualisierungsprofile konfiguriert werden. Ein Visualisierungsprofil speichert die aktuellen Repository-Einstellungen, das heißt die dort definierte Auswahl an Visualisierungselementen ("Installierte Elemente"), die aus einer oder mehreren Bibliotheken stammen. Diese Elemente werden dann in der Werkzeug-Box des Visualisierungs-Editors bereitgestellt, wenn das Profil im Projekt verwendet wird. Um ein Profil zu konfigurieren, wählen Sie die gewünschten Elemente aus der Liste "Verfügbare Elemente" aus, die sich abhängig von den gerade installierten Element-Bibliotheken zusammensetzt. Es kann ein bestehendes Profil modifiziert oder ein neues angelegt werden.

Profilauswahl

Das **Profil**-Feld enthält eine Auswahlliste mit den Namen der bereits definierten Visualisierungsprofile.

Wenn Sie ein bestehendes Profil bearbeiten wollen, wählen Sie dessen Namen aus der Liste und verwenden den Dialog "Verfügbare Elemente", um - wie unten beschrieben - die Liste "Installierte Elemente" zu füllen. Die neue Konfiguration wird unter dem Profilnamen gespeichert, sobald der Dialog über Schaltfläche Schließen verlassen wird.

Wenn Sie ein neues Profil anlegen wollen, verwenden Sie **Hinzufügen** oder **Kopieren**. In jedem Fall erscheint der Dialog **Name für Visualisierungsprofil eingeben** zum Definieren des neuen Profilnamens. Standardmäßig wird automatisch der zuvor verwendete Name eingetragen, erweitert um "_0", um eindeutig zu werden. Sie können statt dessen jedoch auch einen beliebigen anderen eindeutigen Namen eingeben:

Abb. 172 Dialog 'Namen für Visualisierungsprofil eingeben'

Wenn Sie ein neues Profil hinzufügen, ist die Liste "Installierte Elemente" zunächst leer. Wenn Sie ein bestehendes Profil kopieren, ist sie mit den entsprechenden Elementen gefüllt..

Installierte Elemente

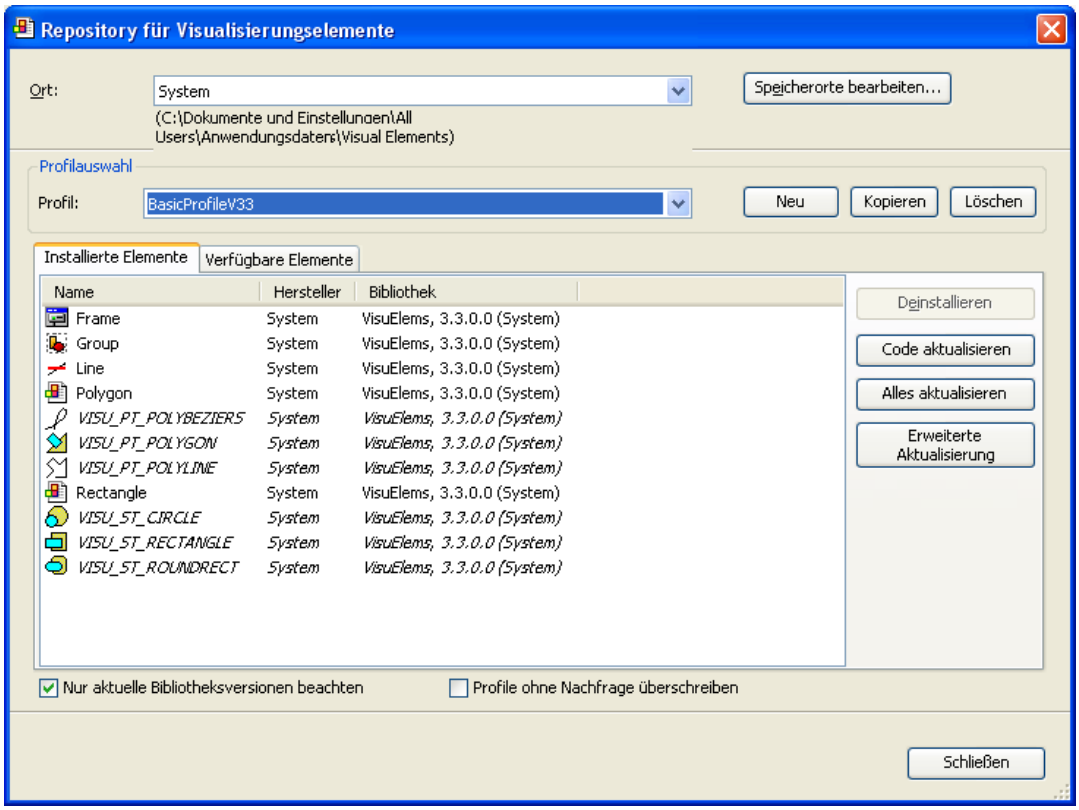


Abb. 173 Dialog 'Repository für Visualisierungselemente', 'Installierte Elemente'

Die Liste zeigt die Elemente, die gerade dem im Profilauswahl-Feld angegebenen Profil zugeordnet sind. Angegeben sind jeweils der **Name**, der **Hersteller** und die zugrundeliegende Elemente-**Bibliothek**.

Sofern Sie nur die jeweils aktuellste Version der Bibliothek durchsuchen und in der Liste berücksichtigen möchten, aktivieren Sie das Kontrollkästchens **Nur aktuelle Bibliotheksversionen beachten**. Daraufhin wird die Operation zeitsparender ausgeführt.

Mit **Deinstallieren** werden alle gerade in der Liste ausgewählten Elemente deinstalliert; das heißt sie werden aus der Liste "Verfügbare Elemente" entfernt.

Die Liste kann über eine der folgenden Schaltflächen aktualisiert werden, beispielsweise nachdem Bibliotheken ausgetauscht wurden:

Code aktualisieren: Die Liste wird hinsichtlich eventueller Änderungen im Implementations-Code der Bibliotheksbausteine aktualisiert.

Alles aktualisieren: Die Liste wird nicht nur hinsichtlich möglicher Änderungen im Implementations-Code, sondern auch bezüglich Änderungen in den Schnittstellen (Deklarationsteilen) der Bibliotheksbausteine aktualisiert.

Erweiterte Aktualisierung: Die Liste wird mit speziellen Elementparametern aktualisiert. Zu diesem Zweck erscheint der Dialog 'Erweiterte Aktualisierung', wo man beispielsweise die zulässige Anzahl von Punkten in Polygon-Elementen verändern kann. Die aktualisierte Einstellung gilt für das gerade bearbeitete Profil.

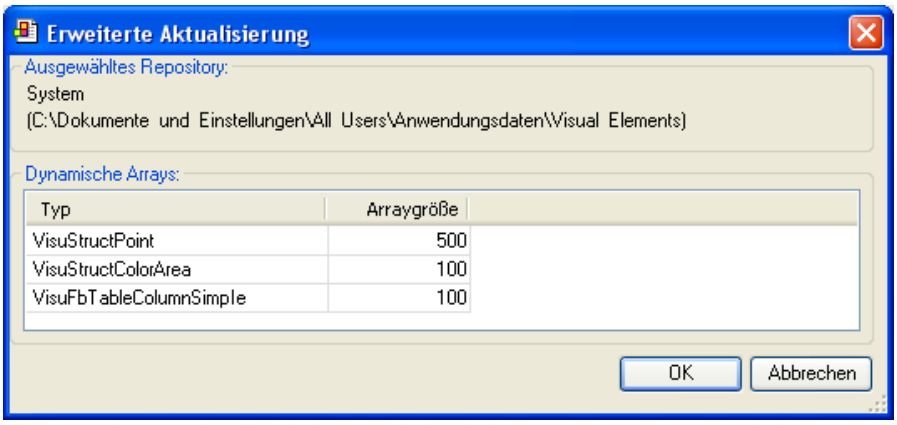


Abb. 174 Dialog 'Erweiterte Aktualisierung

Um weitere Elemente zu installieren öffnen Sie das Registerblatt 'Verfügbare Elemente':

Verfügbare Elemente

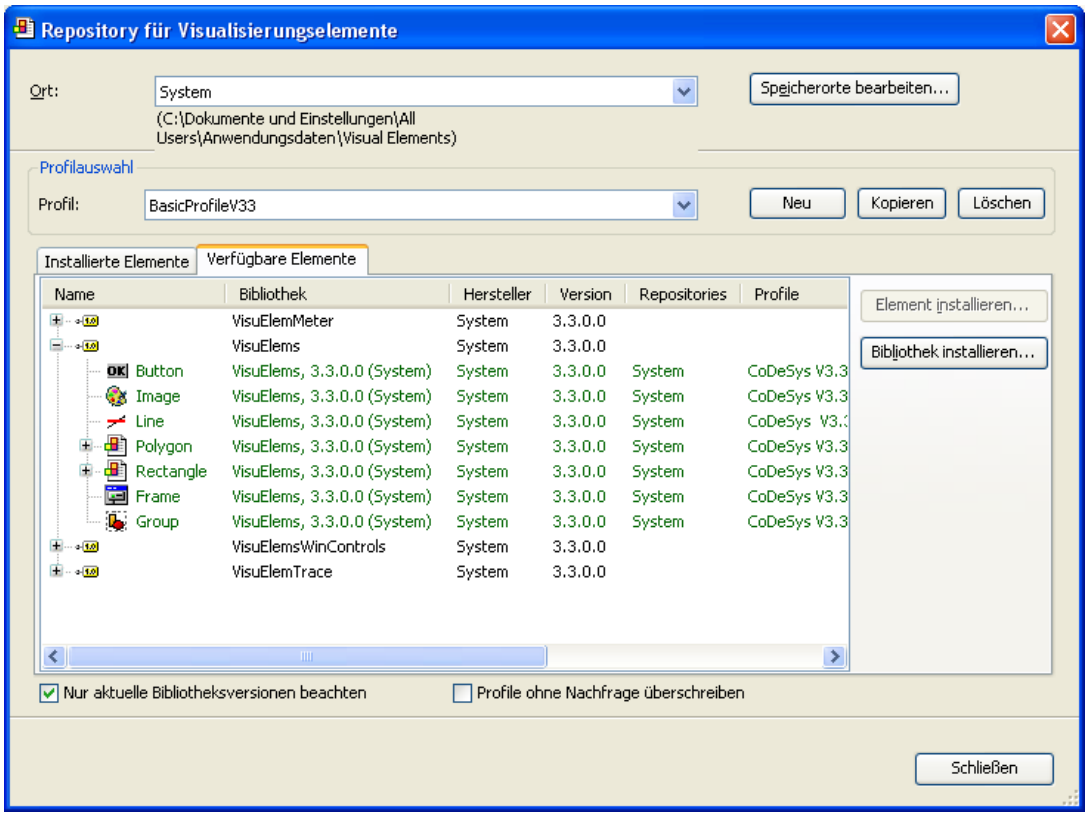


Abb. 175 Dialog 'Repository für Visualisierungselemente', 'Verfügbare Elemente'

Welche Elemente aktuell auf dem System zur Verfügung stehen, hängt von den gerade installierten Bibliotheken mit Visualisierungselementen ab.

Die Liste zeigt in einer Baumstruktur die im Repository liegenden Bibliotheken und ihre Elemente. **Grün** dargestellte Elemente sind bereits für das im Profilauswahl-Feld angegebene Profil installiert.

Um eine weitere Bibliothek zu installieren und damit deren Elemente verfügbar zu machen, öffnen Sie über Schaltfläche **Bibliothek installieren...** den Bibliotheks-Repository Dialog,wo die Installation durchgeführt werden kann. Die neue Bibliothek wird daraufhin in der Liste der Verfügbaren Elemente hinzugefügt.

Um ein oder mehrere Elemente für die Verwendung mit dem angegebenen Profil zu installieren, wählen Sie es oder sie in der Liste aus und bedienen Schaltfläche **Element installieren...** Das bzw. die Elemente werden daraufhin der Liste 'Installierte Elemente' hinzugefügt.

Hinweis: Derzeit kann nur eine einzige Version eines Elements installiert werden. Wenn Sie versuchen zwei Versionen eines Elements zu installieren, wird eine Fehlermeldung ausgegeben.

12.19 Bibliotheksverwalter

Diese Befehlskategorie enthält Befehle zum Arbeiten im Bibliotheksverwalter, also für den Umgang mit den im Projekt verwendeten Bibliotheken.

12.19.1 Befehle der Kategorie Bibliotheksverwalter

Das Plug-In 'Bibliotheksverwalter' stellt die unten genannten Befehle für die Kategorie 'Bibliotheksverwalter' bereit. Sie können aus dem Editorfenster des Bibliotheksverwalters aufgerufen werden, der "Bibliothek hinzufügen"-Befehl standardmäßig auch aus dem Menü 'Bibliotheken', das in der Menüleiste erscheint, sobald der Editor aktiv ist. Wenn erforderlich, kann die Menükonfiguration über den Anpassen-Dialog verändert werden.

- Bibliothek hinzufügen...
- Eigenschaften...
- Bibliothek erneut laden

12.19.2 Bibliothek hinzufügen...

Dieser Befehl (Kategorie 'Bibliotheksverwalter') ist standardmäßig im Menü 'Bibliotheken' und im Bibliotheksverwalter-Fenster verfügbar.

Verwenden Sie den Befehl, wenn eine Bibliothek über den gerade aktiven Bibliotheksverwalter ins Projekt eingefügt werden soll. Nur in einem lokalen Repository installierte Bibliotheken können eingefügt werden. Es ist möglich, mehrere Versionen einer Bibliothek gleichzeitig im Projekt zu verwenden, über einen oder mehrere Bibliotheksverwalter.

Der Befehl öffnet den Dialog 'Bibliothek hinzufügen' mit den Registerblättern '**Bibliothek**' und '**Platzhalter**'.

Teildialog 'Bibliothek'

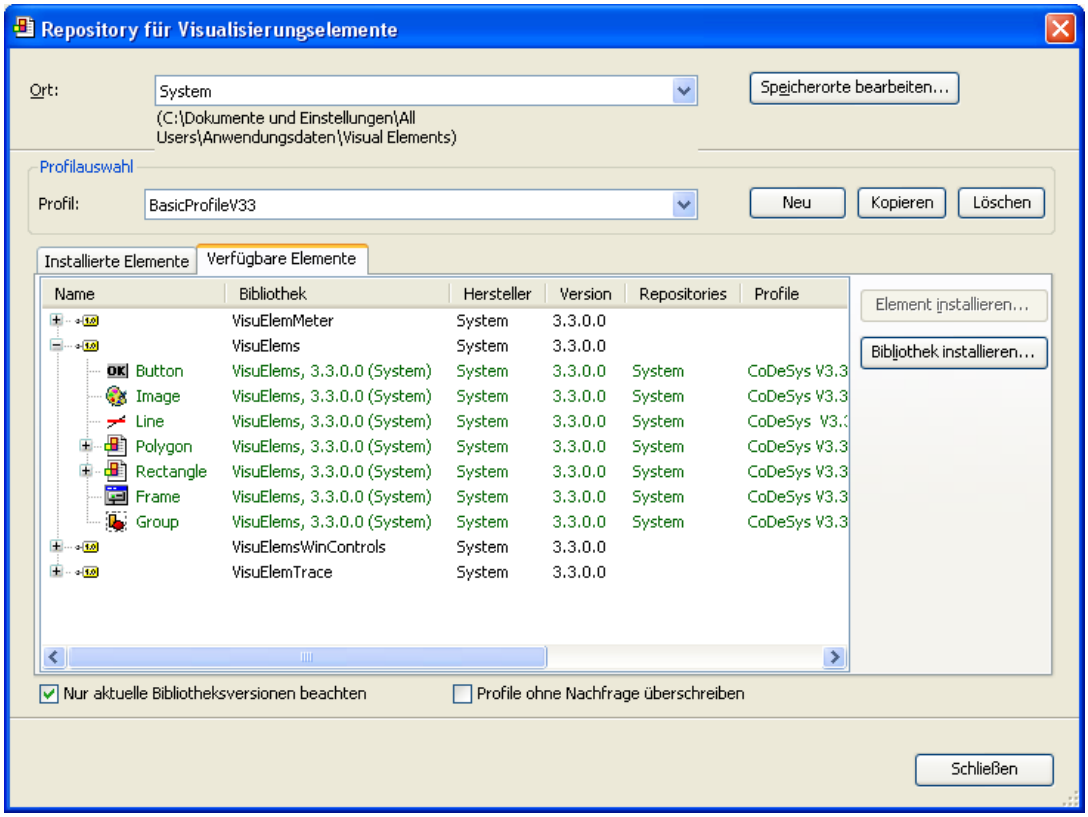


Abb. 176 Dialog 'Bibliothek hinzufügen', 'Bibliothek', Nur letzte Version anzeigen

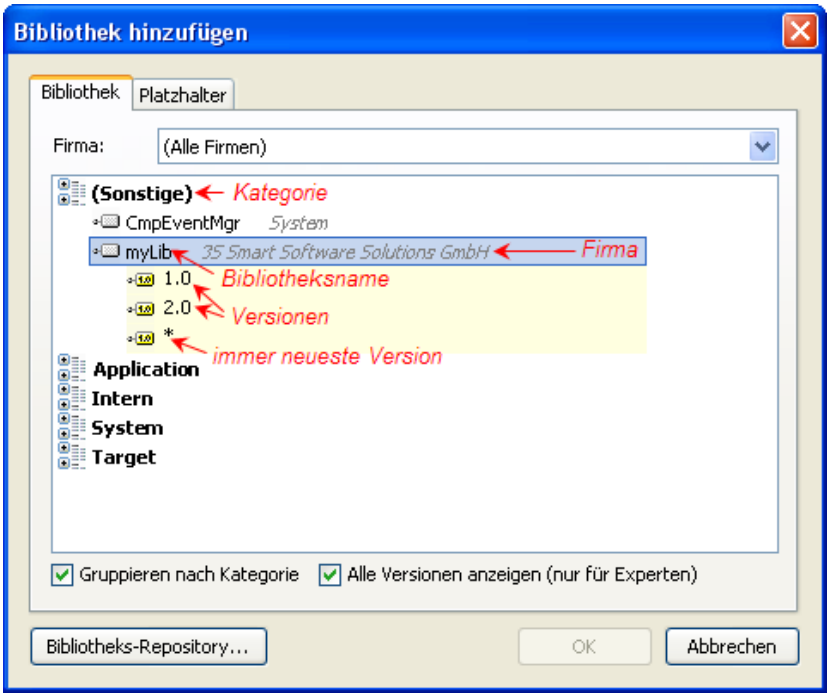


Abb. 177 Dialog 'Bibliothek hinzufügen', 'Bibliothek', Alle Versionen anzeigen

Hier werden **alle installierten Bibliotheken** angeboten. Die Liste kann nach **Firma** gefiltert werden. "(All companies)" zeigt alle verfügbaren Bibliotheken).

Wenn die Option **Gruppieren nach Kategorie** aktiviert ist, werden die Bibliotheken der gerade eingestellten Firma gemäß der definierten Kategorien gruppiert. Die Kategorienbezeichnungen erscheinen als Knoten, ein Klick auf einen Knoten öffnet die Liste der zugehörigen Bibliotheken (**Bibliotheksnamen**) bzw. Unterkategorien, ein Klick auf einen Bibliotheksnamen die verfügbaren **Versionen**. Wenn nicht nach Kategorien gruppiert wird, erscheint die Liste aller Bibliotheksnamen alphabetisch sortiert.

Wählen Sie die gewünschte Bibliothek aus. Wenn die Option **Alle Versionen anzeigen (nur für Experten)** aktiviert ist, werden alle aktuell installierten Versionen einer Bibliothek aufgelistet. Zusätzlich erscheint jeweils ein "*", was "**aktuellste im Repository verfügbare Version**" bedeutet. In diesem Fall können Sie also auch unter den Versionen wählen. Allerdings ist die Option standardmäßig ausgeschaltet, so dass nur die aktuellste Version dargestellt wird. In diesem Fall ist eine **Mehrfachauswahl** von Bibliotheken möglich: Halten Sie dazu die <Umschalt>-Taste gedrückt, während Sie die gewünschten Bibltioheken auswählen.

Nach Bestätigen der Auswahl mit OK wird die Bibliotheksversion im Bibliotheksverwalter eingefügt.

Wenn Sie eine Bibliothek einfügen wollen, die noch nicht auf dem lokalen System installiert ist, können Sie über die Schaltfläche **Bibliotheks-Repository** direkt den gleichnamigen Dialog öffnen, um die Installation durchzuführen.

Teildialog 'Placeholder'

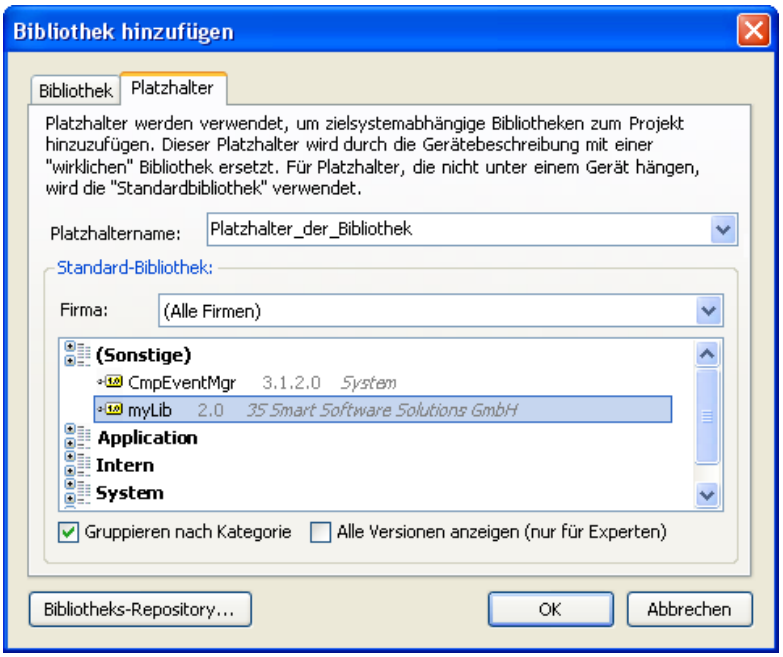


Abb. 178 Dialog 'Bibliothek hinzufügen', 'Platzhalter'

Der Reiter 'Platzhalter' kommt in folgenden Anwendungen zum Tragen:

- Erzeugung eines vom Zielgerät unabhängigen Projekts
- Anlegen eines Bibliotheksprojekts <library_xy>, das eine weitere zielgerätspezifische Bibliothek referenziert

Platzhalter innerhalb eines Projekts

Soll ein Projekt kompatibel für mehrere austauschbare Zielgeräte sein, so müssen die gerätespezifischen Bibliotheken via Platzhalter in den Bibliotheksverwalter des Projekts eingebunden werden.

Sobald ein Zielgerät spezifiziert wird, werden die Platzhalter gemäß der zugehörigen Gerätebeschreibung besetzt. Doch auch wenn aktuell noch keine Gerätebeschreibung zur Verfügung steht, ermöglichen die Platzhalter schon einen syntaktischen Test des Projekts.

Um eine Bibliothek per Platzhalter in den Bibliotheksverwalter einzufügen, muss sie zunächst aus der im Abschnitt **Standard-Bibliothek** angebotenen Liste ausgewählt werden. Dabei kann die Auswahl durch Nennung einer **Firma** eingeschränkt werden.

Des weiteren muss der **Platzhaltername** im entsprechenden Feld eingegeben werden. Um eine fehlerfreie Eingabe zu erleichtern, werden in der Auswahlliste alle aktuell in Gerätebeschreibungen definierten Platzhalternamen angeboten.

Platzhalter innerhalb eines Bibliotheksprojektes

Basiert ein Bibliotheksprojekt auf anderen Bibliotheken, welche abhängig sind vom Zielgerät, so müssen diese Bibliotheken per Platzhalter in das Bibliotheksprojekt eingebunden werden.

Das heißt, es wird hier nicht eine konkrete Bibliothek eingebunden, sondern nur ein Platzhalter. Dieser Platzhalter wird später, wenn <library_xy> in einem Projekt für ein bestimmtes Gerät verwendet wird, durch den Namen der gerätespezifischen Bibliothek ersetzt. Dieser muss allerdings in der zugehörigen **Gerätebeschreibungsdatei** definiert sein. Dort muss ein Eintrag vorliegen, der dem Platzhalternamen den Namen einer vorliegenden "reellen" Bibliothek zuweist.

Wenn aus irgendeinem Grund keine gerätespezifische Bibliothek verfügbar ist, wird der Platzhalter mit der **Default library** ersetzt, die hier im Dialog angegeben ist. (Dies ermöglicht beispielsweise eine fehlerfreie Übersetzung des Bibliotheksprojekt, auch wenn zu diesem Zeitpunkt keine geeignete Gerätebeschreibung vorliegt.)

Im Feld **Platzhalter Name** geben Sie eine Zeichenfolge als Namen für den Platzhalter ein. Wählen Sie außerdem eine der aktuell installierten Bibliotheken als **Standard-Bibliothek** aus. Gehen Sie dazu vor wie beim Einfügen einer Bibliothek, siehe oben bei Unterdialog 'Bibliothek'. Wie dort kann die Option **Alle Versionen anzeigen (nur für Experten)** aktiviert werden, um alle aktuell auf dem System installierten Versionen einer Bibliothek angezeigt zu bekommen.

Nach Schließen des Dialogs mit OK wird die "Platzhalter-Bibliothek" im Bibliotheksverwalter-Baum eingefügt. Wenn der Eigenschaften-Dialog für diesen Eintrag geöffnet wird, erhält man darin im Feld 'Standard' Information über die dafür definierte Standard-Bibliothek.

Im folgenden Beispiel ist der Platzhalter noch nicht ersetzt; wenn der Bibliotheksverwalter später unterhalb eines Gerätes mit entsprechender Gerätebeschreibung hängt, erscheint im Feld 'Name' an seiner Stelle der Name der gerätespezifischen Bibliothek.

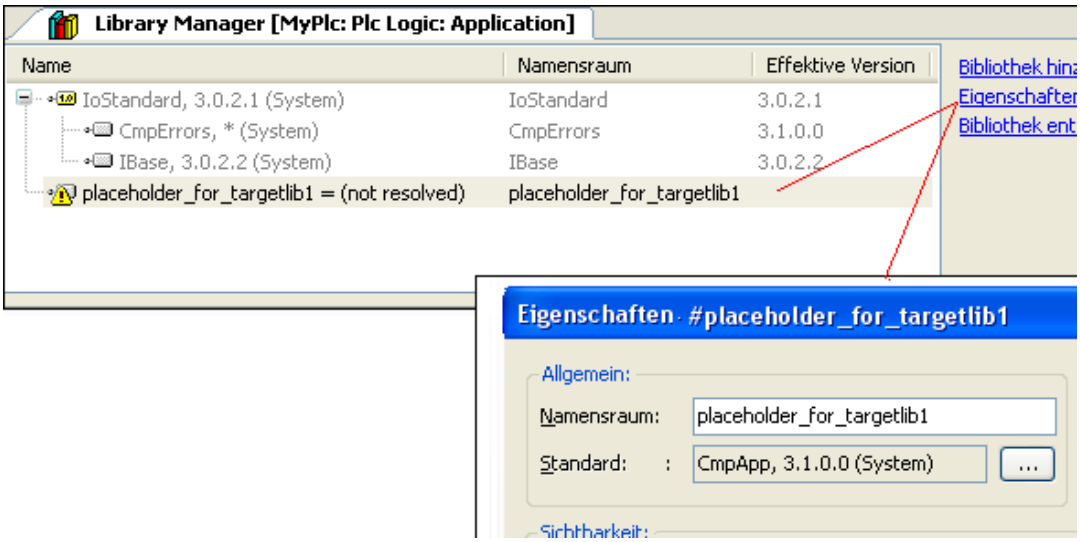


Abb. 179 Beispiel für einen Bibliotheks-Platzhalter im Bibliotheksverwalter

12.19.3 Eigenschaften...

Dieser Befehl (Kategorie 'Bibliotheksverwalter') ist im Editorfenster des Bibliotheksverwalters verfügbar, wenn gerade ein Bibliothekseintrag selektiert ist.

Er öffnet den **Eigenschaften** Dialog für die ausgewählte Bibliothek und erlaubt die folgenden Einstellungen bezüglich Namensraum, Versionshandling, und Verhalten für den Fall, dass die Bibliothek in einer anderen referenziert und über diese in ein Projekt eingebunden wird:

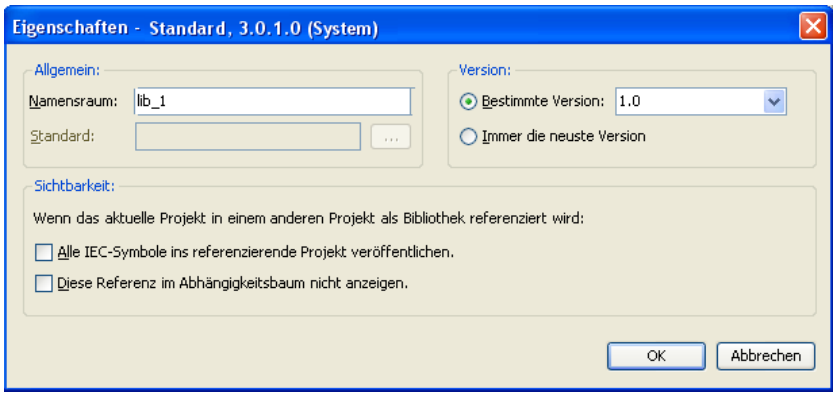


Abb. 180 Eigenschaften-Dialog einer Bibliothek

Allgemein:

Namensraum: Der aktuelle Namensraum der Bibliothek wird angezeigt. Standardmäßig ist er identisch mit dem Bibliotheksnamen, außer es wurde beim Erstellen der Bibliothek explizit ein anderer Standard-Namensraum in den Projektinformationen definiert. Der Namensraum kann für das lokale Projekt jederzeit hier im Eigenschaften-Dialog geändert werden.

Standard: Wenn gerade ein Bibliotheks-Platzhalter im Bibliotheksverwalter ausgewählt ist, enthält dieses Feld den Namen der Bibliothek, die den Platzhalter ersetzen soll, wenn kein gerätespezifische Bibliothek verfügbar ist.

Version:

Hier kann konfiguriert werden, welche Version der Bibliothek verwendet werden soll:

Bestimmte Version: Genau diese Version wird verwendet. Wählen Sie die gewünschte von der Liste..

Immer die neueste Version: Es wird immer die neueste im Bibliotheks-Repository gefundene Version der Bibliothek verwendet. Das heißt, die tatsächlich verwendeten Bibliotheksmodule können sich ändern, weil eine neuere Version der Bibliothek verfügbar ist.

Sichtbarkeit: Diese Einstellungen sind von Interesse, sobald die Bibliothek in einer anderen eingebunden, d.h. dort referenziert wird. Standardmäßig sind sie deaktiviert:

Alle IEC-Symbole ins referenzierende Projekt veröffentlichen. :Solange diese Option deaktiviert ist, können die Bausteine der Bibliothek, wenn diese später in einer anderen eingebunden ist, eindeutig angesprochen werden, indem der entsprechende Namensraum-Pfad verwendet wird. Dieser setzt sich dann zusammen aus dem Namensraum der "Vater"-Bibliothek und dem eigenen und wird dem Bausteinnamen vorangestellt.

Hinweis: Empfehlung: Die Option sollte nur dann aktiviert werden, wenn Sie eine sog. "Container-Bibliothek" erstellen wollen, d.h. eine Bibliothek, die keine eigenen Bausteine definiert, sondern nur andere Bibliotheken referenziert, um eine Art Bibliotheksbündel zu erstellen. Dies kann sinnvoll sein, um mehrere Bibliotheken gleichzeitig in ein Projekt einzubinden, einfach indem die Container-Bibliothek eingebunden wird. In diesem Fall ist es dann allerdings wünschenswert, die einzelnen Bibliotheken des Pakets top-level im Bibliotheksverwalter des Projekts zu platzieren, um die Zugriffspfade für die Bibliotheksbausteine zu verkürzen. Eben dies kann durch Aktiveren der vorliegenden Option vorausblickend erzielt werden.

Diese Referenz im Abhängigkeitsbaum nicht anzeigen: Wenn diese Option aktiviert wird, wird die Bibliothek später, wenn sie - eingebunden in eine andere Bibliothek - über diese in ein Projekt eingefügt wird, im Bibliotheksverwalter nicht angezeigt. Das erlaubt das Einfügen **"versteckter" Bibliotheken**, erfordert aber bewusstes Vorgehen, da es im Fall von Übersetzungsfehlern, die auf Bibliotheksfehler zurückgehen, ev. Probleme beim Finden der Ursache geben kann.

12.19.4 Bibliothek erneut laden

Dieser Befehl (Kategorie 'Bibliotheksverwalter') ist im Editorfenster des Bibliotheksverwalters verfügbar, wenn eine Bibliothek ausgewählt ist, deren Laden beim Öffnen des Projekts fehlgeschlagen ist.

Wenn eine Bibliothek aus irgendeinem Grund nicht im definierten Repository-Pfad bereitsteht, wenn ein Projekt geöffnet wird, wird eine entsprechende Fehlermeldung ausgegeben.

Nachdem Sie den Fehler behoben haben, also die Bibliothek wieder ordnungsgemäß verfügbar ist, wählen Sie den Befehl 'Bibliothek erneut laden' für diese Bibliothek. Somit kann die Bibliothek nachgeladen werden, ohne dass das Projekt verlassen werden muss.

12.20 Meldungsfenster

Die Befehlskategorie 'Meldungsfenster' enthält Befehle zum Navigieren zwischen Meldungen im Meldungsfenster oder zwischen Meldungen und Implementierungs-Code.

12.20.1 Befehle der Kategorie Meldungsfenster

Das Plug-In 'MessageView' stellt für Kategorie 'Message View' Befehle bereit, die dem Navigieren zwischen im Meldungsfenster ausgegebenen Meldungen bzw. zwischen den Meldungen und den betreffenden Stellen im Projekt dienen.

Standardmäßig sind diese Befehle im 'Bearbeiten'-Menü enthalten. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog umkonfiguriert werden.

- Gehe zu Quelltextposition
- Nächste Meldung
- Vorherige Meldung

12.20.2 Gehe zu Quelltextposition

Wenn der Cursor in einer Zeile des Meldungsfensters steht, die einen Fehler oder eine Warnung anzeigt, kann mit diesem Befehl (Kategorie Meldungsfenster) zur entsprechenden Stelle im Projekt (Quelltext) gesprungen werden, die die Meldung verursacht hat. Der Cursor wird dann dort platziert. Wenn das betroffene Objekt noch nicht in einem Fenster geöffnet war, wird es jetzt geöffnet.

Um zur nächsten bzw. vorherigen Meldung im Meldungsfenster zu gelangen, verwenden Sie die Befehle Nächste Meldung bzw. Vorherige Meldung.

12.20.3 Nächste Meldung

Tastenkürzel: <F4>

Verwenden Sie diesen Befehl (Kategorie Meldungsfenster), um den Cursor in der nächsten Meldung im Meldungsfenster zu platzieren. Vom Ende der Liste wird zurück an den Anfang gesprungen.

12.20.4 Vorherige Meldung

Tastenkürzel: <Umsch> + <F4>

Verwenden Sie diesen Befehl (Kategorie Meldungsfenster), um den Cursor in der vorangehenden Meldung im Meldungsfenster zu platzieren. Vom Anfang der Liste wird zurück ans Ende gesprungen.

12.21 Objekte

Die Befehlskategorie'Objekte' enthält Befehle für die Handhabung von Projektobjekten und Ordnern im Geräte- und POUs-Fenster.

12.21.1 Object Commands

Befehle der Kategorie Objekte - Object Commands

Das Plug-In 'ObjectCommands' stellt Befehle der Kategorie 'Objekte' für das Handhaben der Objekte in einem Projekt bereit. Standardmäßig sind diese Befehle im 'Projekt' Menü enthalten. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog verändert werden.

Die Befehle:

- Objekt hinzufügen
- Objekt bearbeiten
- Objekt bearbeiten mit....
- Objekt bearbeiten - Offline
- Schnittstellen implementieren...
- Exportieren...
- Importieren...
- Aktive Applikation setzen

Objekt hinzufügen

Symbol:

Dieser Befehl (Kategorie Objekte) öffnet ein Untermenü mit den Objekten, die an der gerade ausgewählten Position im POUs- oder Gerätebaum eingefügt werden können. Beispielsweise kann ein Applikations-Objekt nur eingefügt werden, wenn gerade ein Geräteobjekt im Gerätefenster ausgewählt ist, eine Aktion kann nur eingefügt werden, wenn gerade ein Programm-Objekt ausgewählt ist.

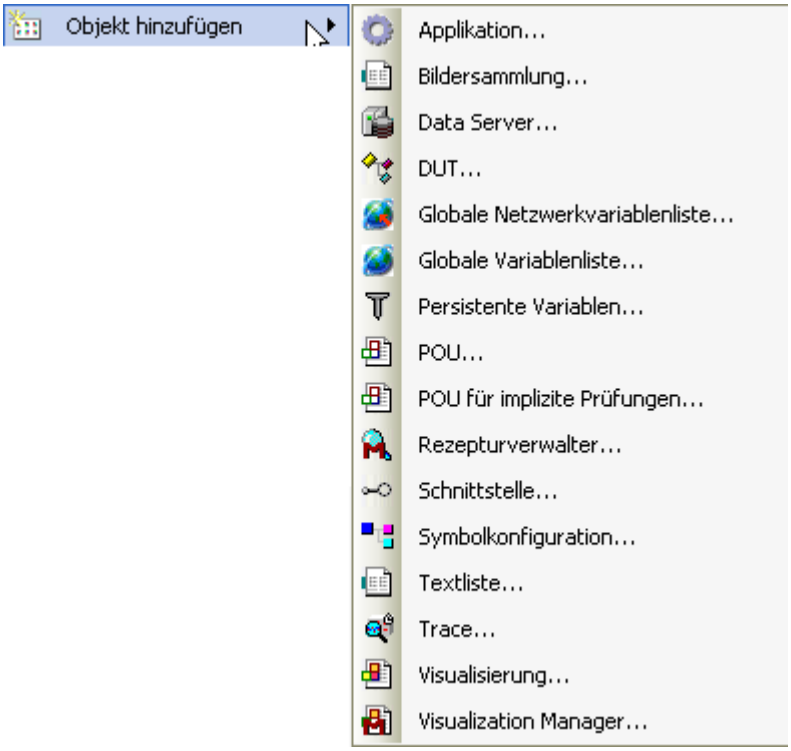


Abb. 181 Menü 'Objekt hinzufügen'

Wählen Sie den gewünschten Objekttyp und definieren im sich öffnenden Dialog einen **Namen** für das neue Objekt im POUs- bzw. Gerätebaum. Bitte beachten Sie dabei die Empfehlungen zur Namensvergabe um eine gewisse Einheitlichkeit zu erreichen. Je nach Objekttyp können auch noch weitere Einstellungen im Dialog vorgenommen werden. Sehen Sie hierzu bitte die Hilfeseiten zum Objekttyp bzw. zum entsprechenden Editor:

Aktion

Applikation

Bibliotheksverwalter

Bildersammlung

Datenquelle

Data Server

DUT

Eigenschaft

Externe Datei
Gerät
Global Variable List
Methode
Persistente Variablen
POU
POUs für implizite Prüfungen
Eigenschaft
Rezepturverwalter
Rezepturdefinition
Schnittstelle
Symbolkonfiguration
Taskkonfiguration
Task
Textliste
Trace
Transition
Visualisierung
Visualization Manager

Um ein Objekt im POUs-Baum **umzubenennen**, klicken Sie auf den Objekteintrag um ein Editierfeld zu öffnen, oder verwenden Sie den Eigenschaften-Dialog (Kontextmenü).

Hinweis: Ein Geräteobjekt muss über den speziellen Befehl: Gerät hinzufügen hinzugefügt werden.

Objekt bearbeiten

Symbol:

Dieser Befehl (Kategorie Objekte) kann verwendet werden, wenn ein Objekt, das im POUs oder Geräte-Fenster vorliegt, zur Betrachtung oder zum Bearbeiten geöffnet werden soll. Wenn das Objekt im Baum ausgewählt ist, ist der Befehl im Kontextmenü oder standardmäßig im Projekt-Menü verfügbar.

Das Objekt wird im entsprechenden Editor geöffnet. Im Online Modus wird ein Dialog geöffnet, in dem nochmals nachgefragt wird, in welcher Ansicht das Objekt geöffnet werden soll. (Dialog 'Online Status auswählen').

Objekt bearbeiten mit...

Zur Zeit arbeitet dieser Befehl wie Objekt bearbeiten.

Objekt bearbeiten - Offline

Dieser Befehl (Kategorie Objekte) ist nur im Online Modus verfügbar. Nach einem Login in die Applikation öffnet ein Doppelklick auf einen Eintrag im Gerätebaum den geeigneten Editor im Offline Modus (sofern eine Instanz des ausgewählten Objekts im Projekt entalten ist). Falls Sie es vorziehen, das Objekt im Offline Modus zu öffnen, so selektieren Sie es durch eienne einfachen Klick auf den Eintrag im Gerätebaum und rufen das Kommando "Objekt bearbeiten (Offline)" aus, das sich standardmäßig im Menü unter dem Punkt "Projekt" findet.

Schnittstellen implementieren...

Dieser Befehl (Kategorie Objekte) wird verwendet, um für einen Funktionsblock die Aktualisierung der implementierten Schnittstellen zu erreichen. Das heißt: Wenn die Schnittstellendefinition verändert wurde, z.B. eine weitere Methode hinzugefügt wurde, kann dies im Funktionsblock nachgezogen werden, indem das Funktionsblock-Objekt im POUs- oder Geräte-Fenster selektiert wird und der Befehl 'Schnittstellen implementieren...' ausgeführt wird.

Beispiel:

1. Schnittstelle ITF1 enthält Methode GetName; Beim Anlegen der Funktionsblöcke FB1 und FB2 wurde ITF1 jeweils implementiert und somit wird GetName auch unterhalb von FB1 und FB2 automatisch eingefügt:

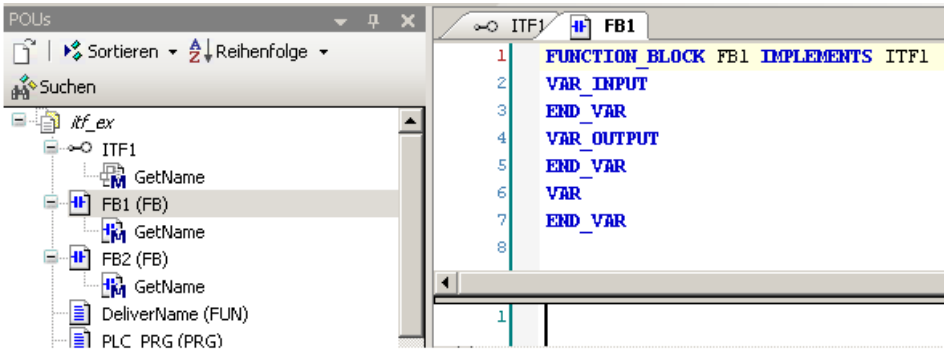


Abb. 182 FB1 und FB2 implementieren ITF1

2. Schnittstelle ITF1 wird nun um Methode GetID erweitert:

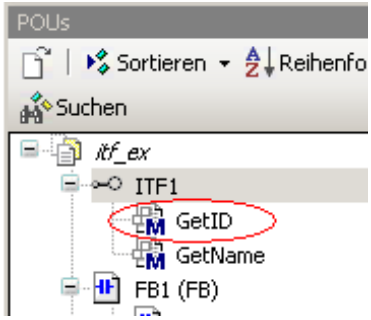


Abb. 183 Schnittstellenänderung

3. Aktualisierung von FB1 bzgl. Schnittstelle ITF1 mittels Befehl 'Schnittstelle implementieren' (Kontextmenü):

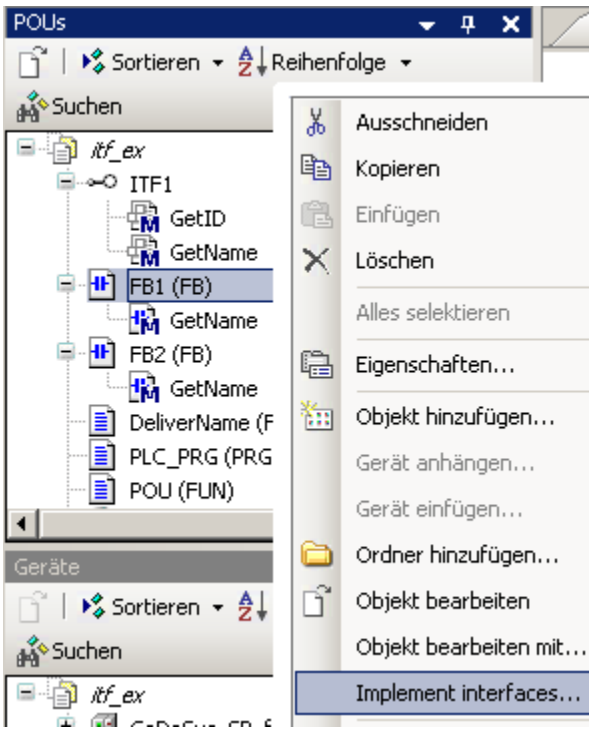


Abb. 184 Aktualisierung der Schnittstelle in FB1

-> Aktualisierte Schnittstellendefinition im Funktionsblock:

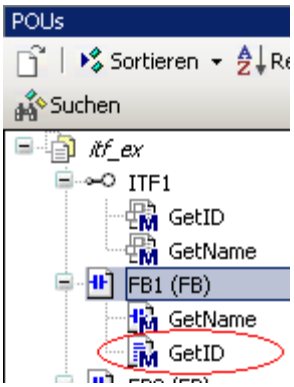


Abb. 185 Schnittstelle in FB1 aktualisiert

Exportieren...

Dieser Befehl (Kategorie Objekte) wird verwendet, um ein Objekt oder einen ganzen Objekte-"Baum" in eine oder mehrere externe Dateien. Exportdateien werden im XML-Format angelegt und tragen standardmäßig die Erweiterung "**:export**".

Dabei gibt es folgende Fälle:

- Wenn der Befehl im **Projekt-Menü** (Standardposition des Befehls) angewählt wird, öffnet sich zunächst der Dialog **Exportieren** zum Auswählen der gewünschten Objekte aus dem POUs- bzw. Geräte-Baum. "Kindobjekte" eines ausgewählten Objekts (z.B. Aktionen unterhalb eines Programm-Objekts) sind standardmäßig mit ausgewählt. Die Auswahl kann jedoch explizit durch Setzen oder Entfernen der entsprechenden Haken verändert werden.
  Wenn die Option **Eine Datei pro Teilbaum** aktiviert ist, werden alle Objekte, die zum selben Unterbaum gehören, in die selbe Exportdatei exportiert, ansonsten wird für jedes Objekt eine eigene Datei erzeugt. Nach Auswahl der gewünschten Objekte erhält man noch einen Dialog zum Definieren des Zielordners für die Exportdatei(en) und - wenn nur eine einzige Exportdatei erzeugt wird - zum Definieren des Dateinamens. Wenn dieser Dialog geschlossen wird, erfolgt der Exportvorgang.

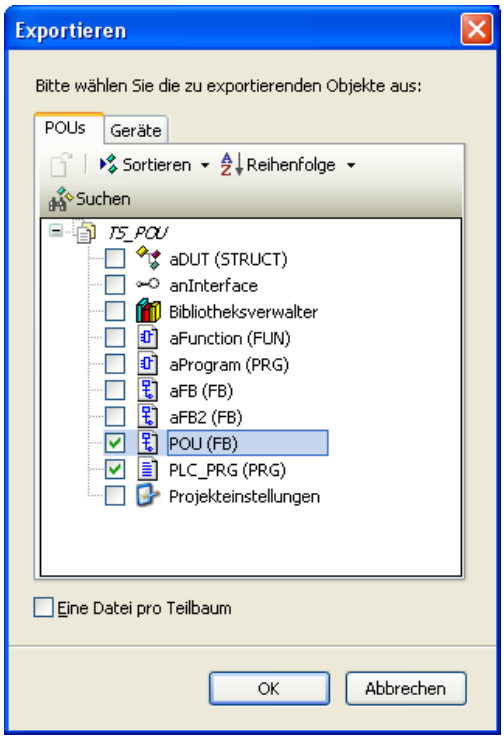


Abb. 186 Dialog 'Exportieren'

- Wenn der Befehl über das **Kontextmenü** ausgeführt wird, öffnet sich ein Dialog zur Auswahl des Zielordners für die Exportdatei. In dem angegebenen Ordner wird dann für jedes Objekt, das gerade im POUs- oder Geräte-Baum ausgewählt ist, eine Exportdatei angelegt. In diesem Fall erhält eine Exportdatei automatisch den Namen des Objekts bzw. des gesamten Pfades des Objekts im POUs- oder Gerätebaum. Beispiele: "Device.export" für ein top-level-Object "Device"; "Device.App1.POU1.export" für ein Objekt "POU1", das unterhalb des Applikationsobjekts "App1" eingefügt ist, welches wiederum unterhalb "Device" hängt. Zu beachten ist, dass beim Auswählen eines Objekts dessen ggfs. vorhandenen "Kindobjekte" (z.B. Aktionen unterhalb eines Programm-Objekts) nicht automatisch mit exportiert werden, wie es bei Verwenden des 'Exportieren'-Dialogs (siehe oben) standardmäßig der Fall ist. Sie müssen explizit ebenfalls ausgewählt werden.

Exportdateien können über den Befehl Importieren... in ein Projekt reimportiert werden.

Importieren...

Dieser Befehl (Kategorie Objekte) wird verwendet, um ein oder mehrere Objekte, die in Form einer Exportdatei im XML-Format außerhalb des Projekts bereitstehen, ins Projekt zu importieren.

Ein Dialog zum Suchen nach Exportdateien wird geöffnet. Standardmäßig ist der Filter auf Dateien mit der Erweiterung "**.export**" gesetzt. (Exportdateien können mit Hilfe des Befehls 'Exportieren' aus Objekten eines Projekts erzeugt werden).

Die importierten Objekte werden dem geöffneten Projekt hinzugefügt. Wenn bereits Objekte mit gleichem Namen existieren, werden entsprechende Meldungsdialoge ausgegeben und es kann ggfs. gewählt werden, ob und welche Objekte überschrieben werden soll oder nicht.

Aktive Applikation setzen

Dieser Befehl (Kategorie Objekte) wird verwendet, um zu definieren, welche der Applikationen, die im **Geräte-Fenster** angelegt sind, die aktive sein soll. Wenn gerade eine Applikation im Gerätebaum selektiert ist, kann der Befehl im Kontextmenü und standardmäßig im Projekt-Menü angewählt werden. Nach Ausführung wird der Objekteintrag der Applikation im Baum fettgedruckt dargestellt.

Beachten Sie, dass Online-Aktionen immer nur die "aktive Applikation" betreffen !

12.21.2 Navigation

Befehle der Kategorie Objekte - Navigation

Das Plug-In 'Navigator Control' stellt Befehle der Befehlskategorie 'Objekte' bereit, die dazu dienen, Ordner in den View-Fenstern zu handhaben. Standardmäßig sind diese Befehle im Menü 'Projekt' enthalten. Wenn erforderlich, kann mit Hilfe des Anpassen-Dialogs die Menüstruktur umkonfiguriert werden.

Die Befehle:

Ordner hinzufügen...

Ordner hinzufügen...

Symbol:

Dieser Befehl (Kategorie Objekte) ist im Kontextmenü verfügbar, wenn gerade im POUs- oder Geräte-Fenster gearbeitet wird.

Wenn ein Ordner in der obersten Ebene des Objektebaums einer der Fenster hinzugefügt werden soll, muss sichergestellt sein, dass das Fenster gerade aktiv ist, aber kein vorhandenes Objekt ausgewählt ist. Wenn ein Ordner auf einer tieferliegenden Ebene eingefügt werden soll, muss ein Objekt auf derjenigen Ebene des Objektebaums ausgewählt sein, unterhalb der der Ordner eingefügt werden soll.

Der Befehl öffnet den Dialog **Ordner hinzufügen**, wo ein Name für den neuen Ordner eingegeben werden kann. Der Name darf Leerzeichen, Ziffern und Sonderzeichen enthalten.

Abb. 187 Dialog 'Ordner hinzufügen'

Nach Bestätigen mit OK wird der Ordner im Baum eingefügt. Vorangestellt wird das Icon .

Wenn der neue Ordner unterhalb eines existierenden Eintrags im Baum eingefügt wurde, kann er über einen Mausklick auf das Pluszeichen, das jetzt vor dem Eintrag erscheint, sichtbar gemacht werden.

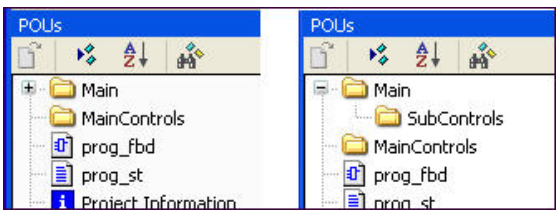


Abb. 188 Beispiel von Ordnern im POUs-Fenster

12.21.3 PLCopenXML

Befehle der Kategorie Objekte - PLCopenXML

Das Plug-In 'PLCopenXML' stellt Befehle der Befehlskategorie 'Objekte' für den Export und Import von Projektobjekten in bzw. aus einer xml-Datei im PLCopen-Format bereit.

Standardmäßig sind diese Befehle im Menü 'Projekt' enthalten. Wenn erforderlich, kann mit Hilfe des Anpassen-Dialogs die Menüstruktur umkonfiguriert werden.

Die Befehle:

- Export PLCopenXML...
- Import PLCopenXML...

Exportiere PLCopenXML...

Dieser Befehl der Kategorie Objekte dient dem Export aller oder einer Auswahl von Objekten aus dem aktuellen Projekt in eine **xml-Datei** gemäß PLCopen-Format.

Der Befehl öffnet den Dialog **Exportiere PLCopenXML**, wo die gewünschten Objekte ausgewählt werden können:

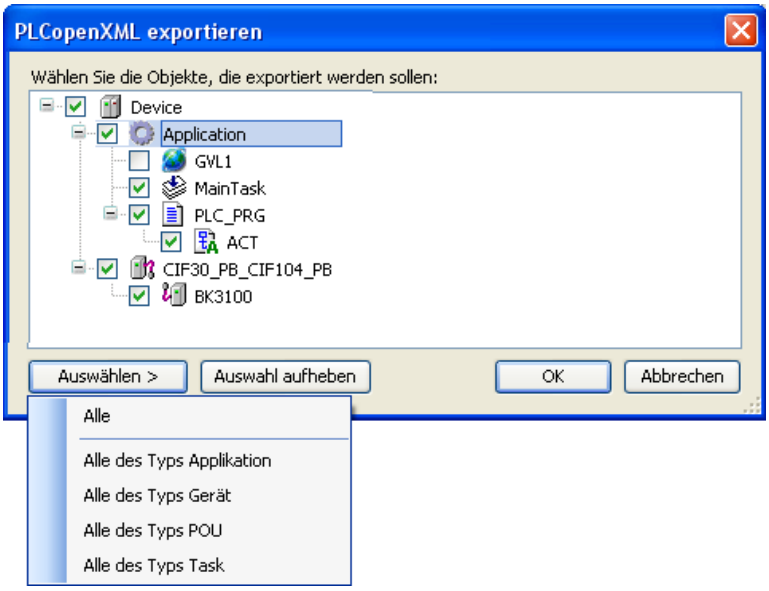


Abb. 189 Exportiere PLCopenXML Dialog

Aus den durch das PLCopen-Format gegebenen Einschränkungen ergeben sich auch Einschränkungen bezüglich der möglichen Auswahl von zu exportierenden Objekten. Sie werden dies feststellen, wenn Sie die gewünschten Objekte im **Export PLCopenXML** Dialog durch einen Mausklick in die voranstehende Box mit einem Haken versehen. Beim Auswählen bestimmter Objekte werden automatisch andere mit ausgewählt werden. Beispielsweise wird auch das Applikationsobjekt mit einem Haken versehen, sobald die zugehörige Taskkonfiguration ausgewählt wird.

Um die Auswahl zu erleichtern, können die Schaltflächen **Auswählen** bzw. **Auswahl aufheben** verwendet werden und eine der angebotenen Optionen verwendet werden. Es gibt die Möglichkeit, gleichzeitig alle Objekte eines bestimmten Objekttyps (Applikation, Gerät, POU, Task) oder alle Objekte überhaupt auszuwählen bzw. deren Auswahl wieder aufzuheben.

Wenn die Auswahl beendet ist, bestätigen Sie mit OK. Der Standarddialog zum Speichern einer Datei wird geöffnet, wo ein Ablageort und Name für die Exportdatei definiert werden kann. Der Dateifilter ist automatisch auf **PLCopen xml Dateien (*.xml)** gesetzt.

Die Exportdatei kann mit Befehl Import PLCopenXML... wieder in ein Projekt importiert werden.

Importiere PLCopenXML...

Dieser Befehl der Kategorie Objekte dient dazu, den Inhalt einer xml-Datei, die Projektobjekte im PLCopen-Format beschreibt, in ein Projekt zu importieren. Die Datei kann aus einem Projekt mit Hilfe des Befehls Exportiere PLCopenXML... erzeugt worden sein.

Der Befehl öffnet den Standarddialog zum Suchen nach einer Datei. Der Datei-Filter ist automatisch auf **PLCopen xml Dateien (*.xml)** gesetzt. Wählen Sie die gewünschte Datei aus und bestätigen mit OK. Der Dialog **PLCopenXML importieren** wird geöffnet:

Registerblatt Inhalt:

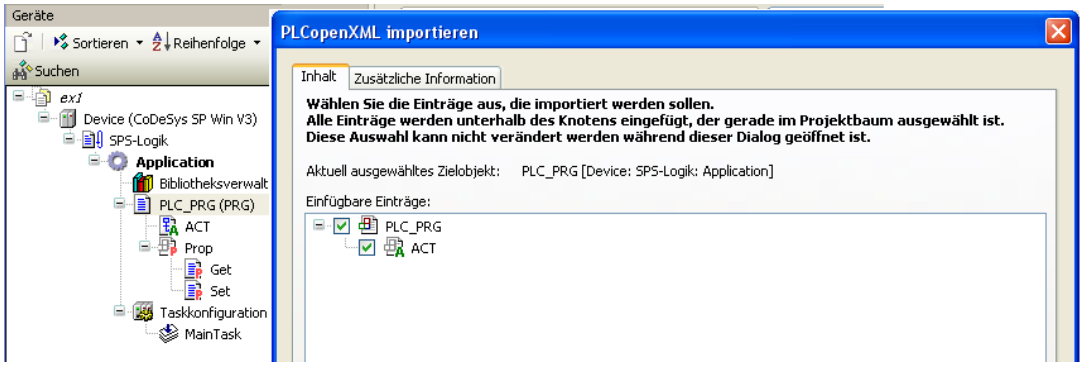


Abb. 190 Dialog 'PLCopenXML importieren', Beispiel

Abhängig vom gerade im POUs oder Gerätebaum ausgewählten Objekt (**Aktuell ausgewähltes Zielobjekt**) listet der Dialog nur diejenigen Objekte bzw. Einträge aus der xml-Datei auf, die an dieser Stelle eingefügt werden können (**Einfügbare Einträge**). Welche davon tatsächlich importiert werden sollen, kann durch das Setzen oder Entfernen des Hakens (soweit es die vorgegebenen Abhängigkeiten zwischen bestimmten Objekt-Typen zulassen) in der dem jeweiligen Eintrag vorangestellten Box konfiguriert werden.

Nach Auswahl der gewünschten Objekte bestätigen Sie mit OK, um sie ins Projekt einzufügen.

Wenn dort bereits ein Objekt gleichen Namens vorhandenist, werden Sie in einem Dialog gefragt, was Sie tun möchten:

- **Neues Objekt umbenennen.** Das Objekt wird unterhalb des bestehenden mit einem neuen Namen "<Name>_<n>" eingefügt, wobei n eine fortlaufende Nummer ist, die beim ersten Import eines gleichnamigen Objekts "1" ist. Beispiel: "PLC_PRG" -> "PLC_PRG_1"
- **Neues Objekt umbenennen. Führe dasselbe für alle nachfolgenden Konflikte aus.** Die Umbenennung wird wie oben für "Neues Objekt umbenennen" beschrieben für das aktuelle Objekt ausgeführt und für alle nachfolgenden Objekte, bei denen der selbe Konflikt auftritt.
- **Bestehendes Objekt ersetzen.** Das bestehende Objekt wird im Projekt durch das importierte ersetzt.
- **Bestehendes Objekt ersetzen. Führe dasselbe für alle nachfolgenden Konflikte aus.** Die Ersetzung wird wie oben für "Bestehendes Objekt ersetzen" beschrieben für das aktuelle Objekt ausgeführt und für alle nachfolgenden Objekte, bei denen der selbe Konflikt auftritt.
- **Neues Objekt überspringen.** Das Objekt wird nicht importiert.
- **Neues Objekt überspringen. Führe dasselbe für alle nachfolgenden Konflikte aus.** Das aktuelle Objekt und alle nachfolgenden, bei denen der selbe Konflikt auftritt, werden nicht importiert.

Zusätzliche Information:

Dieses Registerblatt zeigt die folgenden Informationen, die aus dem File Header und Content Header der PLCopen xml-Datei gelesen werden:

File Header: Firma, Firmen-URL, Produktname, Produktversion, Produktfreigabe, Erstellungsdatum, Inhaltsbeschreibung

Content Header: Name (Projektname), Version, Änderungsdatum, Organisation, Autor, Sprache, Kommentar

12.22 Online

Die Befehlskategorie 'Online' enthält Befehle für das Arbeiten im Online-Modus.

12.22.1 Befehle der Kategorie Online

Die Befehle der Kategorie 'Online' werden vom Plug-In 'Online Commands' bereitgestellt. Sie dienen der Kontrolle des Applikationsprogramms nach dem Einloggen im Online-Betrieb auf einem Zielgerät, das eine reale oder nur simulierte Steuerung sein kann.

Standardmäßig sind die Befehle in den Menüs 'Online' und 'Debug' verfügbar. Wenn erforderlich, kann die Menüstruktur im Anpassen-Dialog verändert werden.

Aus Sicherheitsgründen wird der Benutzer beim Aufruf der Befehle, die unten in der Auflistung mit * markiert sind, grundsätzlich dazu aufgefordert, nochmals zu bestätigen. Zusätzlich wird eine solche Bestätigungsaufforderung für die mit ** markierten Befehle erscheinen, wenn die Option "Sicherer Online-Betrieb" in den Kommunikationseinstellungen des jeweiligen Geräts aktiviert ist.

WARNUNG!
Das außerordentliche Ändern von Variablenwerten in einer auf der Steuerung laufenden Applikation kann zu unerwünschtem Verhalten der gesteuerten Anlage führen.
Evaluieren Sie mögliche Gefahren vor einem Schreiben oder Forcen von Variablenwerten und treffen Sie entsprechende Sicherheitsvorkehrungen.
Abhängig von der gesteuerten Anlage könnten Schäden an der Anlage und Werkstücken entstehen oder Gesundheit und Leben von Personen gefährdet sein.

WARNING!
Online Change verändert das laufende Applikationsprogramm und bewirkt keinen Neustart.
Stellen Sie sicher, dass der neue Applikations-Code dennoch das gewünschte Verhalten des gesteuerten Systems bewirkt.
Abhängig von der gesteuerten Anlage können Schäden an Anlage und Werkstücken entstehen oder Gesundheit und Leben von Personen gefährdet sein.

Die Befehle (* und ** sind oben erklärt):

- Einloggen
- Ausloggen
- Bootapplikation erzeugen
- Laden *
- Online Change *
- Quellcode auf verbundene Steuerung schreiben
- Mehrfacher Download **
- Reset Warm *
- Reset Kalt *
- Reset Ursprung *
- Simulation

Erstellen Menü:

- Start **
- Stop **
- Einzelzyklus **
- Werte schreiben **
- Werte forcen **
- Forcen aufheben
- Alle Forces zum Überwachen-Fenster hinzufügen
- Darstellung

Datei Menü:

- Quellcode-Download...
- Quellcode aus Steuerung laden...

Nicht standardmäßig in einem Menü (siehe Anpassen-Dialog)

- Verbinden mit Gerät
- Trennen von Gerät
- Werte forcen [alle Applikationen]
- Werte schreiben [alle Applikationen}
- Reset Ursprung Gerät
- Forceliste aufheben **

12.22.2 Einloggen

Standard Tastenkürzel: <Alt> + <F8>

Dieser Befehl (Kategorie Online) verbindet das Programmiersystem, d.h. die Applikation mit dem Zielsystem (Steuerung oder simuliertes Gerät) und stellt somit den **Online Modus** her. Die Kommunikationseinstellungen des Geräts müssen dazu korrekt konfiguriert sein und die Applikation muss frei von Kompilierungsfehlern sein (siehe unten).

Für das Laden der aktiven Applikation muss diese fehlerfrei kompiliert worden sein (siehe unten: Übersetzungsvorgang vor dem Einloggen) und die Kommunikationseinstellungen des Geräts müssen korrekt konfiguriert sein.

Falls die Kommunikationseinstellungen nicht passen, erscheint eine Dialogbox, in der gefragt wird, ob der aktive Pfad über eine der folgenden Optionen gesetzt werden soll (oder das Einloggen abgebrochen):

- Option1 (Ja): Der Kommunikationseinstellungen-Dialog soll geöffnet und automatisch ein Netzwerk-Scan durchgeführt werden. Wenn das zuletzt als "aktiver Pfad" eingestellte Gerät gefunden wird (diese Information wird lokal auf Ihrem System gespeichert), soll dieses automatisch als das aktive gesetzt werden.
- Option 2 (Nein): Der Kommunikationseinstellungen-Dialog soll ohne weitere automatisch durchgeführte Aktionen geöffnet werden. Der aktive Pfad muss dann manuell konfiguriert werden.

Wenn der "Einloggen"-Befehl aus dem Online-Menü aufgerufen wird, ist die gerade **aktive Applikation** betroffen.

Wenn der Befehl aus dem Kontextmenü aufgerufen wird, wenn gerade eine Applikation im Gerätebaum selektiert ist, wird mit dieser **ausgewählten Applikation** eingeloggt, auch wenn sie nicht als "aktive Applikation" gesetzt ist.

Hinweis: Über den Objekteigenschaften-Dialog kann auf der Steuerung Speicher für die Applikation reserviert werden.

Folgende Situationen sind beim Einloggen mit der gerade aktiven Applikation (fehlerfrei, Kommunikationseinstellungen korrekt) denkbar:

- **Die Applikation ist noch nicht auf der Steuerung**: Eine Bestätigung des Downloads (Laden) wird angefordert. Dazu erscheint ein Dialog mit folgendem Text: "Applikation <Applikationsname> existiert nicht auf der Steuerung. Soll die Applikation angelegt und geladen werden?" Schaltfläche **Details** in diesem Dialog führt zu Informationen über bereits auf der Steuerung vorhandenen Applikationen (entsprechend der Information im Applikationen-Dialog des Geräteeditors).
- **Die Applikation liegt bereits auf der Steuerung** und wurde seit dem letzten Laden **nicht verändert**. Das Einloggen erfolgt ohne weitere Interaktion mit dem Anwender.
- **Eine andere Version der Applikation liegt bereits auf der Steuerung**, **läuft** aber gerade **nicht**. Es wird gefragt, ob diese ersetzt werden soll. Dazu erscheint ein Dialog mit folgendem Text: "Auf der Steuerung befindet sich eine unbekannte Version der Applikation <Applikationsname>. Mächten Sie diese durch die Applikation im Projekt ersetzen?". Schaltfläche **Details** in diesem Dialog führt zu Informationen über die Applikation im IDE (integrated development environment = Programmiersystem) im Vergleich zu der auf der Steuerung.
- **Eine Version der Applikation liegt bereits auf der Steuerung und läuft**. Es wird gefragt, ob trotzdem eingeloggt und die gerade laufende Applikation überschrieben werden soll. Dazu erscheint ein Dialog mit folgendem Text: "Warnung: Eine unbekannte Version von Applikation <Applikationsname> läuft gerade auf der Steuerung. Möchten Sie dennoch den aktuellen Code laden und die bestehende Applikation ersetzen?" Schaltfläche **Details** in diesem Dialog führt zu Informationen über die Applikation im IDE (integrated development environment = Programmiersystem) im Vergleich zu der auf der Steuerung.
- **Die Applikation liegt bereits auf der Steuerung**, wurde aber seit dem letzten Laden **verändert**. Es wird gefragt, ob ein Online Change durchgeführt werden soll, oder ein Laden der gesamten Applikation, oder ein Einloggen ohne Ändern der gerade laufenden Applikation. Dazu erscheint ein Dialog mit folgendem Text: "Der Code hat sich seit dem letzten Download geändert. Was möchten Sie tun? - MIt Online Change einloggen. - Mit Download einloggen. - Ohne Änderung einloggen." Schaltfläche **Details** in diesem Dialog führt zu Informationen über die im Programmiersystem geänderte Applikation, im Vergleich zu ihrer Vorgängerversion, die gerade auf der Steuerung liegt.

Zusätzlich: Wenn bereits eine oder mehrere **Applikationen aus einem oder mehreren Projekten auf der Steuerung** liegen, erscheint ein Dialog, in dem festgelegt werden kann, ob welche davon vor dem Laden der aktuellen Applikation von der SPS gelöscht werden sollen (entsprechend der Funktionalität des Applikationen-Dialogs des Geräteeditors).

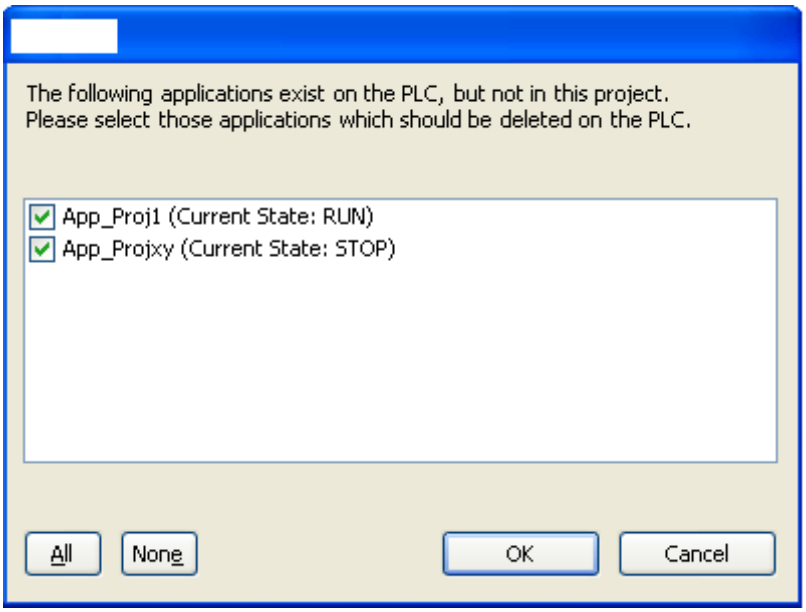


Abb. 191 Meldungsbox, Applikationen auf der Steuerung

Hinweis: Kind-Applikationen (beispielsweise für die Symbolkonfiguration, wenn vom Zielsystem unterstützt, oder für den Trace), die schon einmal auf die Steuerung geladen wurden, während eines später erfolgten Ausloggens jedoch innerhalb des Gerätebaums gelöscht werden, werden bei einem erneuten Einloggen auf das Gerät keinen Online Change auslösen. Zumindest wird jedoch eine Nachfrage erfolgen, ob Sie sie auch auf dem Gerät entfernen wollen.

Übersetzungsvorgang vor dem Einloggen

Vor dem Einloggen und wenn das **aktuelle Applikationsprojekt seit dem Öffnen oder seit seiner letzten Änderung noch nicht kompiliert wurde**, wird es übersetzt. Das bedeutet, es wird wie bei einem "Übersetzen"-Lauf im ausgeloggten Zustand kompiliert und es wird auch Kompilierungscode für die Steuerung erzeugt.

Wenn während der Kompilierung Fehler auftreten, erscheint eine Meldungsbox, die Sie darüber mit folgendem Text informiert: "Es gab Übersetzungsfehler. Möchten Sie sich einloggen ohne das Programm zu laden?" So können Sie wählen, zunächst die Fehler zu korrigieren oder trotz Fehler auf den letzten Stand der eventuell bereits auf der Steuerung vorhandenen Applikation einzuloggen.

Die Fehler werden im Meldungsfenster in Kategorie "Übersetzen" ausgegeben. Siehe hierzu : Übersetzen

Informationen zum Ladevorgang

Wenn das Projekt beim Einloggen ganz, oder bei Online Change teilweise auf die Steuerung geladen wird, erscheinen im Meldungsfenster Informationen zur generierten Codegröße, zur Größe der globalen Daten, zum resultierenden Speicherbedarf auf der Steuerung, und im Fall von Online Change eine Liste der betroffenen Bausteine.

12.22.3 Ausloggen

Standard Tastenkürzel: <Strg> + <F8>

Dieser Befehl (Kategorie Online) beendet ein Ausloggen der Applikation, d.h. die Verbindung des Entwicklungssystems mit dem Zielsystem (Steuerung oder simuliertes Gerät) wird beendet und in den in den **Offline Modus** zurückgekehrt.

Wenn der Befehl aus dem Online-Menü aufgerufen wird, ist die gerade **aktive Applikation** betroffen.

Wenn der Befehl aus dem Kontextmenü aufgerufen wird, wenn gerade eine Applikation im Gerätebaum selektiert ist, wird mit dieser **ausgewählten Applikation** eingeloggt, auch wenn sie nicht als "aktive Applikation" gesetzt ist.

12.22.4 Bootapplikation erzeugen

Dieser Befehl (Kategorie Online) ist sowohl im Online-Modus als auch im Offline-Modus verfügbar, um ein Bootprojekt zu erzeugen. Eine Bootapplikation (Bootprojekt) wird automatisch gestartet, wenn die Steuerung gestartet wird. Sie erhält den Namen <Applikationsname>.app.

Gebrauch im Online-Modus:

Die Bootapplikation wird unter dem Namen **<Applikationsname>.app** auf dem Zielgerät gespeichert.

Gebrauch im Offline-Modus:

Es öffnet sich der Standarddialog zur **Abspeicherung einer Datei**. Wählen Sie ein Verzeichnis zur Abspeicherung der aktuellen Applikation als Bootapplikation in Dateiformat um sie zu einem späteren Zeitpunkt auf ein Zielsystem laden zu können. Die Datei ist vom Typ "Bootapplikationen" und ihrem Namen wird automatisch die Endung ".app" hinzugefügt. Nach Bestätigung mit **Speichern** erscheint ein weiterer Dialog, in welchem Sie gefragt werden, ob ein evtl. im Projektverzeichnis schon vorhandenes Übersetzungsprotokoll überschrieben werden soll:

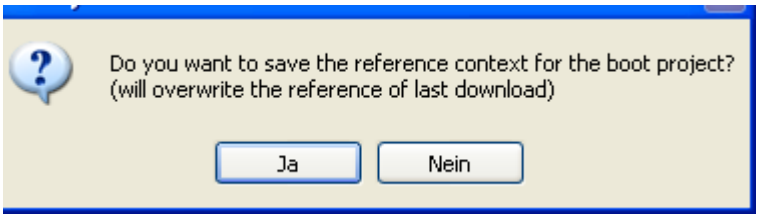


Abb. 192 Dialog zum Speichern der Übersetzungsinformationen für die Bootapplikation

Wählen Sie **Ja**, falls Sie beabsichtigen das neue Bootprojekt mit Hilfe eines externen Tools auf die Steuerung zu übertragen und Sie sich dennoch in die Applikation einloggen wollen ohne zwingend neu herunterladen zu müssen. Ein Übersetzungsprotokoll von einem vorhergehenden Herunterladen würde nämlich nicht zu dem neu erstellten Projekt passen, da es Code und Referenzen der vorhergehenden Applikation enthält.

Hinweis für »PLC Designer« SP Win V3 Geräte:

Um die Bootapplikation auf die Steuerung zu laden, ist es nicht nur erforderlich, sie dorthin zu kopieren, sondern es müssen auch die folgenden Einträge in der Konfigurationsdatei (*.cfg) des Geräts enthalten sein: Name der Applikation, Initialisierung des Retain-Bereichs (nicht für Symbolapplikationen).

```
[CmpApp]
...
Application.1=MyApplication
Application.2=MyApplication.__Symbols
MyApplication.FirstLoadInitRetains=1
```

12.22.5 Laden

Dieser Befehl (Kategorie Online) ist im Online Modus verfügbar. Er bewirkt ein Übersetzen und ein Kompilieren der aktiven Applikation. Das heißt, dass neben einer Syntax-Prüfung auch Applikations-Code erzeugt und auf die Steuerung geladen wird. Im selben Zug wird im Projektverzeichnis das Übersetzungsprotokoll **<projectname>.<devicename>.<application ID>.compileinfo** erzeugt.

Hinweis: Alle Variablen mit Ausnahme von persistenten Variablen werden neu initialisiert.

Hinweis: Über den Objekteigenschaften-Dialog kann auf der Steuerung Speicher für die Applikation reserviert werden.

Bitte sehen Sie die Beschreibung des Befehls Einloggen zu den möglichen Situationen um Einloggen und Laden (Download)

Wenn versucht wird, eine Applikation zu laden, während die gleiche Version dieser Applikation bereits auf der Steuerung liegt, erscheint ein Dialog mit der Information "Programm ist unverändert. Applikation wurde nicht geladen". Die Applikation wird also nicht auf die SPS geladen.

Im Meldungsfenster erscheint beim Laden in der Meldungskategorie "Übersetzen" ein Protokoll der ablaufenden Aktionen (Erzeugen des Codes, Durchführen der Initialisierungen etc.), sowie Angaben zu den Speicherbereichen, zur Größe des Codes, der globalen Daten und des allokierten Speichers. Im Gegensatz zum Laden bei Online Change werden zum Zweck der Übersichtlichkeit ab V3.4.0.0 die geänderten Bausteine nicht mehr aufgelistet.

12.22.6 Online Change

⚠WARNUNG!
Online Change verändert das laufende Applikationsprogramm und bewirkt keinen Neustart.
Stellen Sie sicher, dass der neue Applikations-Code dennoch das gewünschte Verhalten des gesteuerten Systems bewirkt.
Abhängig von der gesteuerten Anlage können Schäden an Anlage und Werkstücken entstehen oder Gesundheit und Leben von Personen gefährdet sein.

Hinweise:
1. Wenn ein Online Change durchgeführt wird, werden die applikations-spezifischen Initialisierungen (Referenzfahrt etc.) nicht ausgeführt, weil die Maschine ihren Status beibehält. Aus diesem Grund könnte der neue Programm-Code nicht den gewünschten Effekt haben.
2. Zeigervariablen behalten ihren Wert aus dem letzten Zyklus. Wenn ein Zeiger auf eine Variable zeigt, die durch den Online Change ihre Größe verändert hat, wird der Wert nicht mehr korrekt geliefert. Stellen Sie sicher, dass Zeigervariablen in jedem Zyklus erneut zugewiesen werden.

Mit diesem Befehl (Kategorie Online) wird explizit ein "Online Change" für die aktive Applikation durchgeführt.

Online Change bedeutet, dass nur die geänderten Teile einer bereits auf der Steuerung laufenden Applikation neu in die Steuerung geladen werden. Dies ist nicht möglich nach einem Alles bereinigen oder Applikation bereinigen! Der Bereinigen-Prozeß löscht die Compile-Informationen (Übersetzungsprotoll), die automatisch bei jedem Kompilierungsvorgang (Übersetzen, Code erzeugen) gespeichert werden und die die Basis für einen Online Change sind.

Beim Laden werden im Meldungsfenster in Kategorie "Übersetzen" neben den üblichen Informationen zum Ladevorgang auch die geänderten Schnittstellen, betroffenen Variablen und alle Bausteine, für die neuer Code generiert wurde, aufgelistet. Falls Speicherorte sich ändern, wird in einer Dialogbox auf mögliche Probleme in Zusammenhang mit Pointern hingewiesen.

Ein Online Change wird automatisch angeboten, wenn mit einer Applikation eingeloggt wird, die bereits auf der Steuerung läuft, aber seit dem letzten Download im Programmiersystem verändert wurde.

12.22.7 Quellcode-Download...

Das Kommando (Kategorie Online) steht standardmäßig im Menüpunkt 'File' zur Verfügung und dient dem Übertragen des aktuellen Projekts in Form einer Archivdatei auf eine auszuwählende Steuerung.

Dabei wird der Fortschritt beim Schreiben und Herunterladen der Archivdatei in der Statusleiste angezeigt. Mit Hilfe von **Abrechen** (unten rechts) kann die Aktion abgebrochen werden.

Standardmäßig trägt die Archivdatei den Namen **archive.prj**. Mittels des Kommandos Datei von Steuerung laden kann sie im Offline-Modus wieder in das Programmiersystem geladen werden..

In den Projekteinstellungen, Kategorie 'Source Download' sind auch die Einstellungen für Zielgerät, Inhalt und Zeitvorgabe beim Herunterladen des Quellcodes auf die Steuerung definiert.

12.22.8 Quellcode aus Steuerung laden...

Das Kommando (Kategorie Online) steht standardmäßig im Menüpunkt 'File' zur Verfügung und dient dem Öffnen eines Projekts, welches in Form einer Archivdatei auf einer Steuerung hinterlegt ist. (Eine solche Archivdatei kann auf der Steuerung beispielsweise mit Hilfe von Quellcode auf Steuerung schreiben erzeugt werden.)

Bei Aufruf des Kommandos wird der Dialog **Gerät auswählen** geöffnet, in welchem der Pfad zur Steuerung im Netzwerk eingetragen werden muss (wie in den Kommunikationseinstellungen). Wählen Sie den passenden Eintrag aus dem Gerätebaum und bestätigen Sie Ihre Wahl mit **OK**.

Daraufhin erscheint der Dialog **Projektarchiv,** worin die Auswahl der zu extrahierenden Inhalte der Archivdatei (archive.prj) getroffen wird ebenso wie die Auswahl der Verzeichnisse, in welche die entpackten Dateien abgelegt werden sollen. Die Bedienung dieses Dialogs erfolgt entsprechend zu Project Archive/Extract Archive. Nach Bestätigung der Auswahl mit **OK** werden die Dateien kopiert. Sofern im angegebenen Pfad schon eine Datei gleichen Namens existiert, erfolgt eine Nachfrage, ob diese überschrieben werden soll.

Nach Abschluss werden Sie gefragt, ob die extrahierte Projekt im Programmiersystem geöffnet werden soll.

12.22.9 Quellcode auf verbundene Steuerung schreiben

Das Kommando (Kategorie Online) steht standardmäßig im Menüpunkt 'File' zur Verfügung und dient dem Übertragen des aktuellen Projekts in Form einer Archivdatei auf die gerade verbundene Steuerung.

Dabei wird der Fortschritt beim Schreiben und Herunterladen der Archivdatei in der Statusleiste angezeigt. Mit Hilfe von **Abrechen** (unten rechts) kann die Aktion abgebrochen werden.

Standardmäßig trägt die Archivdatei den Namen **archive.prj**. Mittels des Kommandos Datei von Steuerung laden kann sie im Offline-Modus wieder in das Programmiersystem geladen werden..

In den Projekteinstellungen, Kategorie 'Source Download' sind auch die Einstellungen für Zielgerät, Inhalt und Zeitvorgabe beim Herunterladen des Quellcodes auf die Steuerung definiert.

12.22.10 Start

Symbol:

Dieser Befehl (Kategorie Online) startet die Applikation auf der Steuerung.

Wenn der Befehl aus dem Online-Menü aufgerufen wird, ist die gerade **aktive Applikation** betroffen.

Wenn der Befehl aus dem Kontextmenü aufgerufen wird, wenn gerade eine Applikation im Gerätebaum selektiert ist, wird mit dieser **ausgewählten Applikation** eingeloggt, auch wenn sie nicht als "aktive Applikation" gesetzt ist.

12.22.11 Stop

Symbol:

Dieser Befehl (Kategorie Online) stoppt die Applikation auf der Steuerung.

Wenn der Befehl aus dem Online-Menü aufgerufen wird, ist die gerade **aktive Applikation** betroffen.

Wenn der Befehl aus dem Kontextmenü aufgerufen wird, wenn gerade eine Applikation im Gerätebaum selektiert ist, wird mit dieser **ausgewählten Applikation** eingeloggt, auch wenn sie nicht als "aktive Applikation" gesetzt ist.

12.22.12 Einzelzyklus

Dieser Befehl (Kategorie Online) bewirkt, dass die aktive Applikation für einen Zyklus ausgeführt wird.

12.22.13 Mehrfacher Download...

Dieses Kommando (Kategorie Online) steht sowohl im Online- als auch im Offline-Modus zur Verfügung. Es bewirkt das Übersetzen und die Code Erzeugung aller im Projekt enthaltenen Applikationen, die dafür vom Benutzer vorgesehen wurden. So wird neben einer syntaktischen Überprüfung dieser Applikationen auch der zugehörige Code erzeugt und auf die jeweilige Steuerung geladen. Im Projektverzeichnis wird für jede zum Download ausgewählte Applikation ein entsprechendes Übersetzungsprotokoll mit dem Namen <projectname>.<devicename>.<application ID>.compileinfo erzeugt.

Nach Betätigen des Kommandos "Mehrfacher Download..." öffnet sich ein Dialogfenster, das eine Liste aller im Projekt enthaltenen Applikationen zur Verfügung stellt:

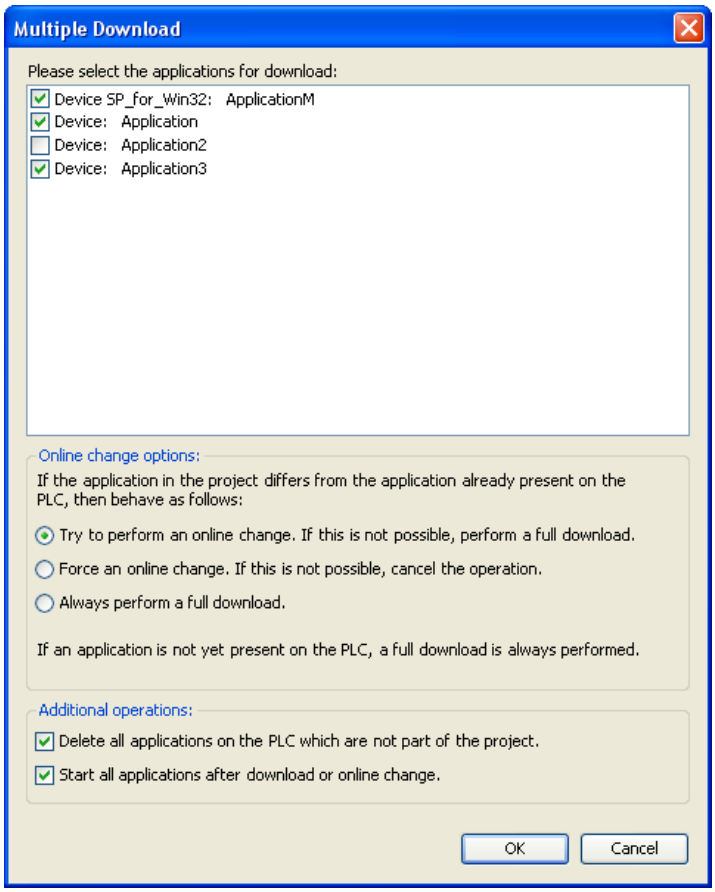


Abb. 193 Dialog "Mehrfacher Download"

Eine Applikation kann durch Aktivieren der zugehörigen Kontrollkästchens für den Download ausgewählt werden. Es dürfen mehrere Applikationen ausgewählt werden, selbst wenn sie auf unterschiedliche Steuerungen geladen werden sollen. Standardmäßig sind entweder alle Kontrollkästchen oder die der vom Benutzer zuletzt gewählten Applikationen aktiviert.

Für den Fall, dass schon eine frühere Version einer ausgewählten Applikation auf der Steuerung existiert und sich von der aktuellen Version unterscheidet, können Sie unter folgenden Optionen wählen:

- **Einen Online Change versuchen. Falls nicht möglich, Download durchführen.** Diese Option ist standardmäßig aktiviert. Bei ihrer Aktivierung werden die modifizierten Teile der ausgewählten Applikationen auf der Steuerung geändert und nur die neu erstellten Teile der betroffenen Applikationen auf die Steuerung geladen.
- **Einen Online Change erzwingen. Falls nicht möglich, Operation abbrechen.** Kann ein Online Change für (zumindest) eine der ausgewählten Applikationen nicht vollzogen werden (z. B. wenn zuvor das Kommando 'Alles bereinigen' oder 'Application bereinigen' ausgeführt wurde), so wird kein Download durchgeführt.

- **Immer einen Download durchführen.** Ohne Berücksichtigung der bereits existierenden Versionen werden alle Teile der ausgewählten Applikationen auf die Steuerung geladen.

Für ausgewählte Applikationen, die noch nicht auf der Steuerung existieren, wird automatisch ein Download auf die zugehörige Steuerung durchgeführt.

Zusätzlich kann durch Aktivierung der entsprechenden Kontrollkästchen bestimmt werden, ob

- alte Applikation, die nicht länger Teil des Projekts sind, von der Steuerung gelöscht werden sollen
- die ausgewählten Applikationen nach Durchführung des Downloads / Online Changes gestartet werden sollen.

Beachten Sie, dass Variablen vom Typ PERSISTENT im allgemeinen nicht initialisiert werden. Falls jedoch das Datenlayout verändert wurde, werden die persistenten Variablen allerdings automatisch neu initialisiert.

Nachdem Sie Ihre Einstellungen im Dialog mit **OK** bestätigt haben, wird zunächst eine syntaktische Überprüfung aller ausgewählten Applikationen durchgeführt. Daraufhin wird für jede Applikation die Kommunikation mit der zugehörigen Steuerung verifiziert, bevor der Download durchgeführt wird.

Nach Abschluss des Downloads erscheint eine Auflistung aller ausgewählten Applikationen, welche detaillierte Informationen über die für jede Applikation durchgeführten Operationen enthält.

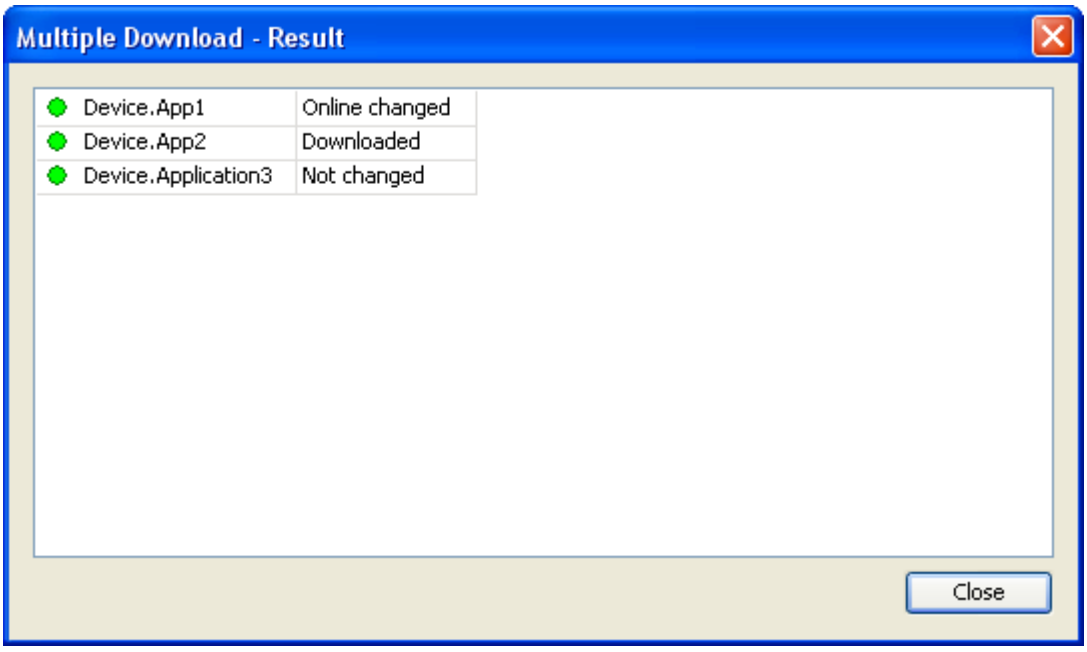


Abb. 194 Dialog "Mehrfacher Download - Ergebnis"

12.22.14 Reset Warm

Dieser Befehl (Kategorie Online) ist im Online-Modus verfügbar. Er setzt – mit Ausnahme der remanenten (Retain, Persistent) -Variablen - alle Variablen der aktiven Applikation auf ihren Initialisierungswert zurück.

Wenn Variablen mit bestimmten Werten initialisiert wurden, werden sie genau auf diese Werte zurückgesetzt. Alle anderen werden auf den Default-Initialisierungswert zurückgesetzt (beispielsweise Integer-Variablen auf 0).

Als Vorsichtsmaßnahme verlangt »PLC Designer« nochmals eine Bestätigung des Anwenders, bevor alle Variablen überschrieben werden. Die Situation entspricht derjenigen bei einem Stromausfall oder beim Aus- und Wiedereinschalten der Steuerung während die Applikation läuft ("Warmstart").

Ein Reset deaktiviert die aktuell im Projekt gesetzten Haltepunkte. Wird das Kommando **Reset Warm** aufgerufen während der Programmablauf gerade auf einem Haltepunkt stoppt, so wird der Benutzer gefragt, ob der aktuelle Zyklus vor Ausführung des Reset beendet werden soll oder ob die Task beendet und der Reset sofort durchgeführt werden soll. Allerdings sind nicht alle Laufzeitsysteme in der Lage einen Reset ohne vorheriges Beenden des aktuellen Zyklus durchzuführen.

Verwenden Sie nach dem Reset den 'Start'-Befehl, um die Applikation wieder zu starten.

Siehe auch: 'Reset (Ursprung)', 'Reset (kalt)'.

12.22.15 Reset Kalt

Dieser Befehl (Kategorie Online) ist im Online-Modus verfügbar. Er entspricht dem Befehl 'Reset' (reset warm), aber neben den "normalen" und Variablen werden auch Retain-Variablen (!) der aktiven Applikation auf ihre Initialisierungswerte zurück gesetzt. Die Situation entspricht der beim Start eines Applikationsprogramms, das gerade auf die Steuerung geladen wurde ("Kaltstart").

Ein Reset deaktiviert die aktuell im Projekt gesetzten Haltepunkte. Wird das Kommando **Reset Kalt** aufgerufen während der Programmablauf gerade auf einem Haltepunkt stoppt, so wird der Benutzer gefragt, ob der aktuelle Zyklus vor Ausführung des Reset beendet werden soll oder ob die Task beendet und der Reset sofort durchgeführt werden soll. Allerdings sind nicht alle Laufzeitsysteme in der Lage einen Reset ohne vorheriges Beenden des aktuellen Zyklus durchzuführen.

Siehe auch 'Reset Wam', 'Reset (Ursprung)'.

12.22.16 Reset Ursprung

Dieser Befehl (Kategorie Online) ist im Online-Modus verfügbar. Er setzt die Werte aller, inclusive der remanenten Variablen der aktiven Applikation auf ihre Initialisierungswerte zurück und **löscht das Applikationsprogramm auf der Steuerung**.

Ein Reset deaktiviert die aktuell im Projekt gesetzten Haltepunkte. Wird das Kommando **Reset Ursprung** aufgerufen während der Programmablauf gerade auf einem Haltepunkt stoppt, so wird der Benutzer gefragt, ob der aktuelle Zyklus vor Ausführung des Reset beendet werden soll oder ob die Task beendet und der Reset sofort durchgeführt werden soll. Allerdings sind nicht alle Laufzeitsysteme in der Lage einen Reset ohne vorheriges Beenden des aktuellen Zyklus durchzuführen.

Siehe auch 'Reset Warm', 'Reset Kalt'.

12.22.17 Werte schreiben

Standard Tastenkürzel: <Strg> + <F7>

WARNUNG!
Das außerordentliche Ändern von Werten in einer auf der Steuerung laufenden Applikation kann zu unerwünschtem Verhalten der gesteuerten Anlage führen.
Evaluieren Sie mögliche Gefahren vor einem Schreiben oder Forcen von Variablenwerten und treffen Sie entsprechende Sicherheitsvorkehrungen.
Abhängig von der gesteuerten Anlage könnten Schäden an der Anlage und Werkstücken entstehen oder die Sicherheit von Personen in Gefahr geraten.

Mit diesem Befehl (Kategorie Online) einen Wert zu "schreiben" bedeutet, die entsprechende Variable in der Steuerung zu Beginn des nächsten Zyklus einmal auf den dafür im Programmiersystem definierten Wert zu setzen. Der Befehl betrifft alle Variablen der **aktiven Applikation**, die für das Schreiben vorbereitet sind.

Hinweis: Befehl "Werte schreiben [alle Applikationen]", der alle Applikationen im Projekt betrifft, ist standardmäßig nicht in einem Menü enthalten. (falls nötig, bitte den Anpassen -Dialog verwenden)

Hinweis: Sehen Sie in diesem Zusammenhang auch den Befehl 'Werte forcen', mit dem Variablenwerte in der Steuerung permanent gesetzt werden können.

Um Variablen für das Schreiben vorzubereiten, muss der gewünschte Wert im Online-Modus an einer der folgenden Stellen, die für Monitoring verwendet werden, definiert werden:

- In einem Überwachen-Fenster, das im Projekt definiert wurde und eine Liste zu überwachender Variablen enthält ("Überwachungsliste", "Watch-Liste").
- In der Online-Ansicht eines Objekts im Deklarationsteil des entsprechenden Editors.

Beispiel:

Öffnen Sie ein Objekt im Online-Modus, z.B. ein Programm, geschrieben in ST. Im Deklarationsteil werden die überwachbaren Ausdrücke (Variablen) in einer Tabelle angezeigt. Klicken Sie auf das entsprechende Feld in Spalte **Vorbereiteter Wert** und geben den gewünschten Wert ein. Führen Sie dann den Befehl **Werte schreiben** aus, der standardmäßig im **Online**-Menü liegt. Der vorbereitete Wert wird daraufhin in der gleichen Zeile in Spalte **Wert** eingetragen, was anzeigt, dass er auf die Steuerung geschrieben wurde. Das Feld 'Vorbereiteter Wert' ist nun wieder leer.

Dasselbe kann in einer Überwachungsliste durchgeführt werden, die den entsprechenden Ausdruck bzw. die Variable enthält.

Beachten Sie in diesem Zusammenhang auch den **Dialog** 'Wert vorbereiten', der für die gerade "geforcten" Variablen geöffnet werden kann, und in dem auch ein neuer Wert zum "Schreiben" definiert werden kann.

12.22.18 Werte forcen, Wert vorbereiten

Standard Tastaturkürzel: <F7>

WARNUNG!
Das außerordentliche Ändern von Variablenwerten in einer auf der Steuerung laufenden Applikation kann zu unerwünschtem Verhalten der gesteuerten Anlage führen.
Evaluieren Sie mögliche Gefahren vor einem Schreiben oder Forcen von Variablenwerten und treffen Sie entsprechende Sicherheitsvorkehrungen.
Abhängig von der gesteuerten Anlage könnten Schäden an der Anlage und Werkstücken entstehen oder Gesundheit und Leben von Personen gefährdet sein.

Dieser Befehl (Kategorie Online) ist im Online-Modus verfügbar. Er bewirkt, dass eine oder mehrere Ausdrücke (Variablen) der **aktiven Applikation** auf der Steuerung permanent auf definierte Werte gesetzt werden. Dieses Setzen erfolgt dann jeweils am Beginn und am Ende eines Abarbeitungszyklus.

Abfolge der Abarbeitung in einem Zyklus: 1.Eingänge lesen, 2. Werte forcen 3. Code abarbeiten, 4. Werte forcen, 5. Ausgänge schreiben.

Hinweis: Befehl "Werte forcen [alle Applikationen]", der alle Applikationen im Projekt betrifft, ist standardmäßig nicht in einem Menü enthalten. (falls nötig, bitte den Anpassen -Dialog verwenden)

Hinweis: Sehen sie auch den Befehl 'Werte schreiben', mit dem ein definierter Wert nur einmal am Beginn des Zyklus gesetzt wird.

Das Forcen wird durchgeführt, bis es explizit durch den Anwender für eine einzelne bzw. für alle Variablen aufgehoben wird, oder bis aus der Applikation ausgeloggt wird.

Um Variablen für das Forcen vorzubereiten, muss der gewünschte Wert im Online-Modus an einer der folgenden Stellen, die für Monitoring verwendet werden, definiert werden:

- In einem Überwachen-Fenster, das im Projekt definiert wurde und eine Liste zu überwachender Variablen enthält ("Überwachungsliste", "Watch-Liste").
- In der Online-Ansicht eines Objekts im Deklarationsteil des entsprechenden Editors.
- In der online-Ansicht eines Objekts innerhalb des Implementierungsteils des FUP /KOP/AWL-Editors.

Ein "geforcter" Wert wird mit einem F Symbol gekennzeichnet.

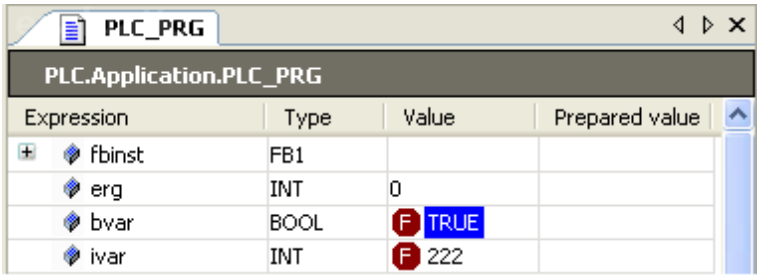


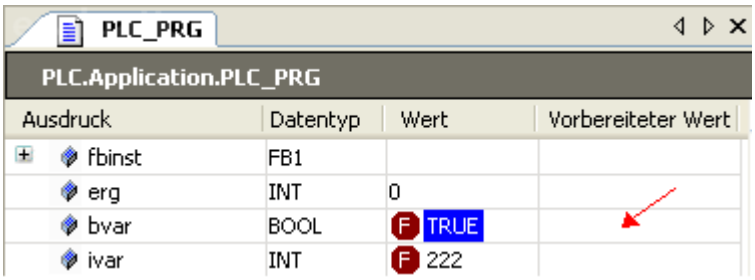
Abb. 195 Beispiel für geforcte Variablen im Deklarationseditor eines Bausteins (Online-Ansicht)

Dialog 'Wert vorbereiten'

Abb. 196 Dialog 'Wert vorbereiten'

Dieser Dialog dient zur Vorbereitung eines Wertes für eine zu forcende Variable, zur Aufhebung eines solchen vorbereiteten Werts, zum Aufheben des Forcens einer Variablen oder zum dabei zusätzlichen Rücksetzen der Variablen auf den Wert, den sie vor dem Forcen innehatte.

Der Dialog öffnet sich, wenn in das Feld 'vorbereiteter Wert' einer gerade geforcten Variable geklickt wird oder in das Inline-Monitoring-Feld der Variablen im Implementierungsteil des FUP/KOP/IL-Editors.



Die folgende Information zur betreffenden Variablen wird angezeigt:

Ausdruck: Pfad: z.B. "PLC.Application.PLC_PRG.ivar"

Datentyp: z.B. "DWORD"

Aktueller Wert: z.B. "TRUE" or "23"

Wählen Sie eine der folgenden Optionen um festzulegen, **was mit der Variable geschehen soll**:

Einen neuen Wert für die nächste Schreib- oder Force-Operation vorbereiten: Abhängig vom Datentyp der Variable kann ein neuer numerischer oder String-Wert eingegeben werden, auf den die Variable gesetzt werden soll.

Vorbereitung mit einem Wert aufheben: Der vorbereitete Wert wird gelöscht.

Den Force aufheben, ohne den Wert zu verändern.: Die Variable wird mit <Unforce> markiert und ist somit dafür vorbereitet, den aktuellen Wert aus der Steuerung zu erhalten.

Den Force aufheben und den Wert auf den Wert zurücksetzen, den er vor der Force-Operation hatte. Die Variable wird mit <Unforce and restore> markiert und ist somit dafür vorbereitet, den Wert zu erhalten, den sie vor der Force-Operation hatte.

Nach Verlassen des Dialogs mit OK zeigt die Variable nun im Feld 'Vorbereiteter Werf' der Monitoring-Ansicht(en) entweder den neu definierten Wert oder "<Unforce>" oder "<Unforce and restore>". Beim nächsten Befehl 'Werte forcen' bzw. 'Werte schreiben' (für die erste Option) werden alle neu vorbereiteten Werte gesetzt.

12.22.19 Forcen aufheben

WARNUNG!
Das außerordentliche Ändern von Variablenwerten in einer auf der Steuerung laufenden Applikation kann zu unerwünschtem Verhalten der gesteuerten Anlage führen.
Evaluieren Sie mögliche Gefahren vor einem Schreiben oder Forcen von Variablenwerten und treffen Sie entsprechende Sicherheitsvorkehrungen.
Abhängig von der gesteuerten Anlage könnten Schäden an der Anlage und Werkstücken entstehen oder Gesundheit und Leben von Personen gefährdet sein.

Dieser Befehl (Kategorie Online) kann im Online Modus verwendet werden, um das Forcen für alle gerade geforcten Variablen der der **aktiven Applikation** aufzuheben.

Die Variablen erhalten dabei ihren aktuellen Wert aus der Steuerung. Dies entspricht Option 'Den Force aufheben, ohne den Wert zu verändern.', die im Dialog Wert vorbereiten für eine geforcte Variable gewählt werden kann.

Hinweis: Befehl "Forcen aufheben [alle Applikationen]", der alle Applikationen im Projekt betrifft, ist standardmäßig nicht in einem Menü enthalten. (falls nötig, bitte den Anpassen -Dialog verwenden)

12.22.20 Alle Forces zum Überwachen-Fenster hinzufügen

Das Kommando (Kategorie Online) steht im Online-Modus zur Verfügung, sobald eine der vier Überwachungsansichten Überwachen 1,Überwachen 2,Überwachen 3 oder Überwachen 4 **aktiv** ist. Es fügt alle aktuell zum Forcen vorbereiteten oder schon geforcten Variablen der aktuellen Applikation zur Watchlist hinzu. Beachten Sie jedoch, dass dies ausschließlich für angedockte Überwachungsansichten funktioniert.

Beachten Sie auch die Überwachungsansicht Watch all Forces, welche stets automatisch alle Variablen zeigt, die zum Forcen vorbereitet wurden und zusätzlich Kommandos bereit stellt um das Forcen aufzuheben.

12.22.21 Forceliste aufheben

WARNUNG!
Das außerordentliche Ändern von Variablenwerten in einer auf der Steuerung laufenden Applikation kann zu unerwünschtem Verhalten der gesteuerten Anlage führen.
Evaluieren Sie mögliche Gefahren vor einem Schreiben oder Forcen von Variablenwerten und treffen Sie entsprechende Sicherheitsvorkehrungen.
Abhängig von der gesteuerten Anlage könnten Schäden an der Anlage und Werkstücken entstehen oder Gesundheit und Leben von Personen gefährdet sein.

Dieser Befehl (Kategorie Online) kann im Online Modus verwendet werden, um das Forcen für alle gerade geforcten Variablen der **aktiven Applikation** aufzuheben.

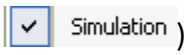
Die Variablen erhalten dabei ihren aktuellen Wert aus der Steuerung. Dies entspricht Option 'Den Force aufheben, ohne den Wert zu verändern.', die im Dialog Wert vorbereiten für eine geforcte Variable gewählt werden kann.

12.22.22 Simulation

Das Kommando (Kategorie Online) steht im Online-Modus zur Verfügung und dient dem An- bzw. Ausschalten des **Simulationsmodus** des Programmiersystems. Im Simulationsmodus kann die aktive Applikation auf einem "simulierten Zielgerät" gestartet und debugged werden. Da solch ein simuliertes Gerät immer in das Programmiersystem integriert ist, wird keine reales Zielgerät benötigt um das Online Verhalten einer Applikation zu testen.

Wenn der Befehl aus dem Online-Menü aufgerufen wird, ist die gerade **aktive Applikation** betroffen.

Wenn der Befehl aus dem Kontextmenü aufgerufen wird, wenn gerade eine Applikation im Gerätebaum selektiert ist, wird mit dieser **ausgewählten Applikation** eingeloggt, auch wenn sie nicht als "aktive Applikation" gesetzt ist.

Nach Aufruf von **Simulation** () wird der Eintrag der Steuerung im Gerätebaum in Kursivschrift dargestellt. Beim ersten Einloggen mit der aktiven Applikation erfolgt die Abfrage, ob die Applikation "Sim.<devicename>.<applicationname>" angelegt und geladen werden soll. Für das simulierte Gerät müssen keine Kommunikationseinstellungen vorgenommen werden. Sehen Sie das Beispiel in folgendem Bild; das Kommando Online \ Einloggen wurde soeben auf die aktive Applikation "ExceptionTest" angewendet:

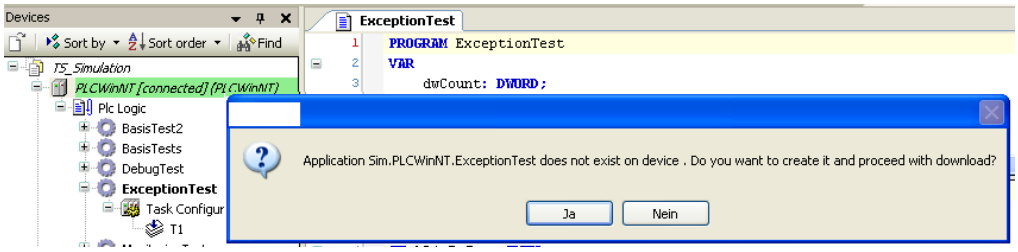


Abb. 197 Beispiel, Login auf simuliertes Gerät

Nach erfolgreichem Einloggen können die entsprechenden Online Kommandos benutzt werden um die Applikation zu testen.

Um den Simulationsmodus auszuschalten loggen Sie sich zunächst aus der Steuerung aus und rufen hernach erneut das Kommando **Simulation** auf. Das dem Kommando vorangestellte Häkchen verschwindet, der Eintrag des Zielgeräts im Gerätebaum wird nicht länger kursiv dargestellt und Sie können sich somit auf das reale Zielgerät einloggen.

12.22.23 Ablaufkontrolle

Standardmäßig ist dieser Befehl (Kategorie Online) im Debug-Menü verfügbar. Er dient dazu, die Ablaufkontrolle ein- und auszuschalten, die für die Spracheditoren ST, FBD, KOP und AWL unterstützt wird.

Die aktivierte Ablaufkontrolle erlaubt die **Abarbeitung des Applikationsprogramms zu verfolgen.**

1. Die aktuellen Werte der Variablen, somit auch die Ergebnisse von Funktionsaufrufen und Operationen werden in den Editorfenstern angezeigt, und zwar an der jeweiligen Abarbeitungsposition und zum jeweiligen Abarbeitungszeitpunkt. Zum Vergleich: Das Standard-Monitioring liefert nur den Wert, den eine Variable zwischen zwei Abarbeitungszyklen hat.
2. Es werden exakt diejenigen Code-Zeilen bzw. Netzwerke farblich markiert, die im aktuellen Zyklus durchlaufen wurden.

Die Ablaufkontrolle **kann nur im Online-Modus aktiviert werden** und arbeitet nur im gerade **sichtbaren** Teil des gerade aktiven Edtiorfensters. 'Ablaufkontrolle aktiviert' wird in der Statuszeile angezeigt, solange die Funktion aktiv ist und im Editorfenster Ablaufkontrollpositionen (durchlaufene Teile des Codes) sichtbar sind.

Hinweis: Aktivierte Ablaufkontrolle verlängert die Laufzeit der Applikation! Wenn die Option Sicherer Online-Betrieb in den Kommunikationseinstellungen aktiviert ist, erscheint beim Einschalten der Ablaufkontrolle eine Meldungsbox. Man wird gefragt, ob die Funktion wirklich aktiviert werden soll und hat die Möglichkeit nochmals abzubrechen.

Darstellung der Ablaufkontrolle in den verschiedenen Spracheditoren:

Standardmäßig ist fahles Grün als Farbe zur Kennzeichnung der Ablaufkontrollpositionen konfiguriert. (Die Farbe kann in den Texteditor-Optionen geändert werden.)

In allen Editoren werden die aktuellen Werte von Variablen und betroffenen Ein- und Ausgängen in kleinen rechteckigen Feldern angezeigt, wie sie auch für das Standard-Monitoring verwendet werden. In durchlaufenen Code-Teilen werden diese Felder in der für die Ablaufkontrolle definierten Farbe dargestellt, in nicht-durchlaufenen erscheinen sie weiss mit grauer Schrift. Man beachte, dass der angezeigte Wert an einer nicht-durchlaufenen Code-Position ein "normaler" Monitoring-Wert ist, d.h. der Wert, der zwischen zwei Taskzyklen vorliegt.

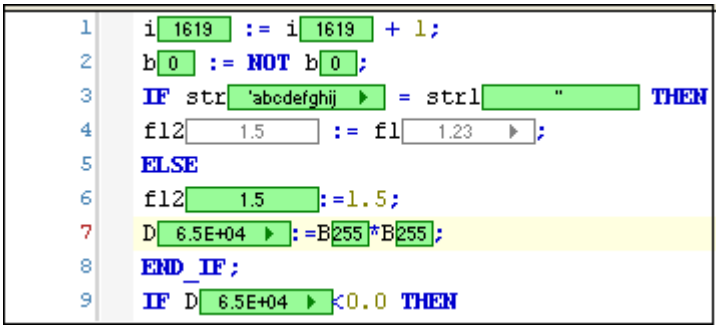


Abb. 198 Beispiel: Ablaufkontrolle im ST-Editor

In Netzwerkeditoren werden die durchlaufenen Netzwerke am linken Rand durch Balken in der 'Ablaufkontroll-Farbe' markiert.

In KOP werden die aktuell durchlaufenen Verbindungslinien grundsätzlich in grün (bzw. der aktuell gesetzten 'Ablaufkontroll-Farbe') dargestellt, die anderen in grau. Der aktuelle Wert der Verbindung wird ebenfalls dargestellt: TRUE durch dicke blaue, FALSE durch dicke schwarze Linien, unbekannte oder analoge Werte durch dünne schwarze. Das kann durch Kombination der jeweiligen Informationen zu gestrichelten Linien führen.

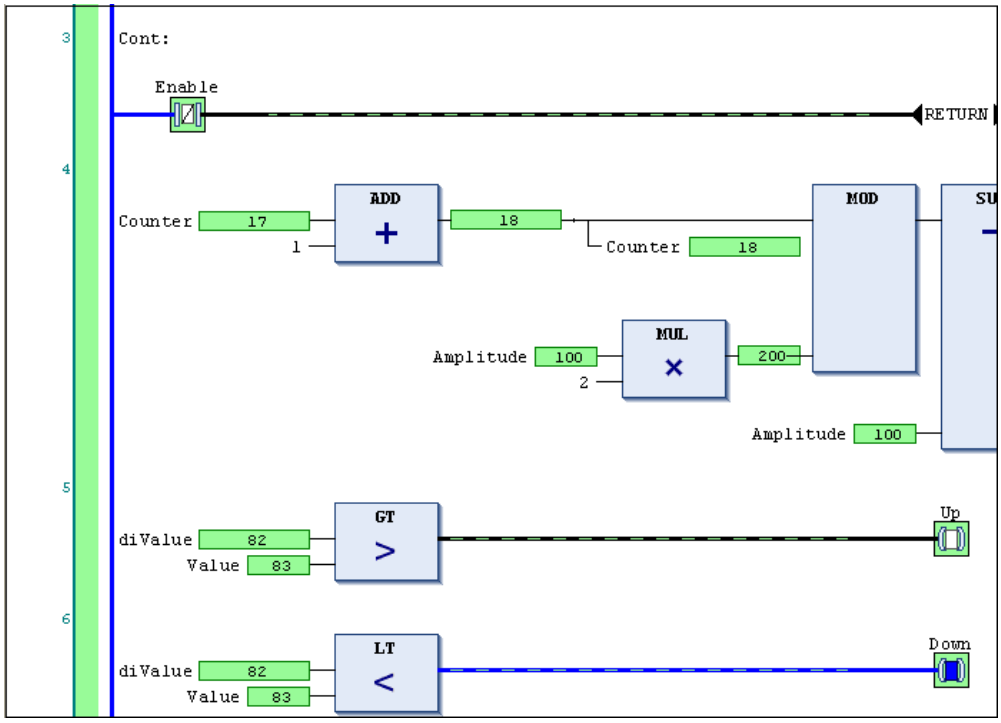


Abb. 199 Beispiel: Ablaufkontrolle im KOP-Editor

In AWL werden für jede Anweisung zwei Felder für die Anzeige der aktuellen Werte verwendet. Eins links des Operators mit dem aktuellen Akkumulator-Wert, eins rechts des Operanden mit dem Operandenwert.

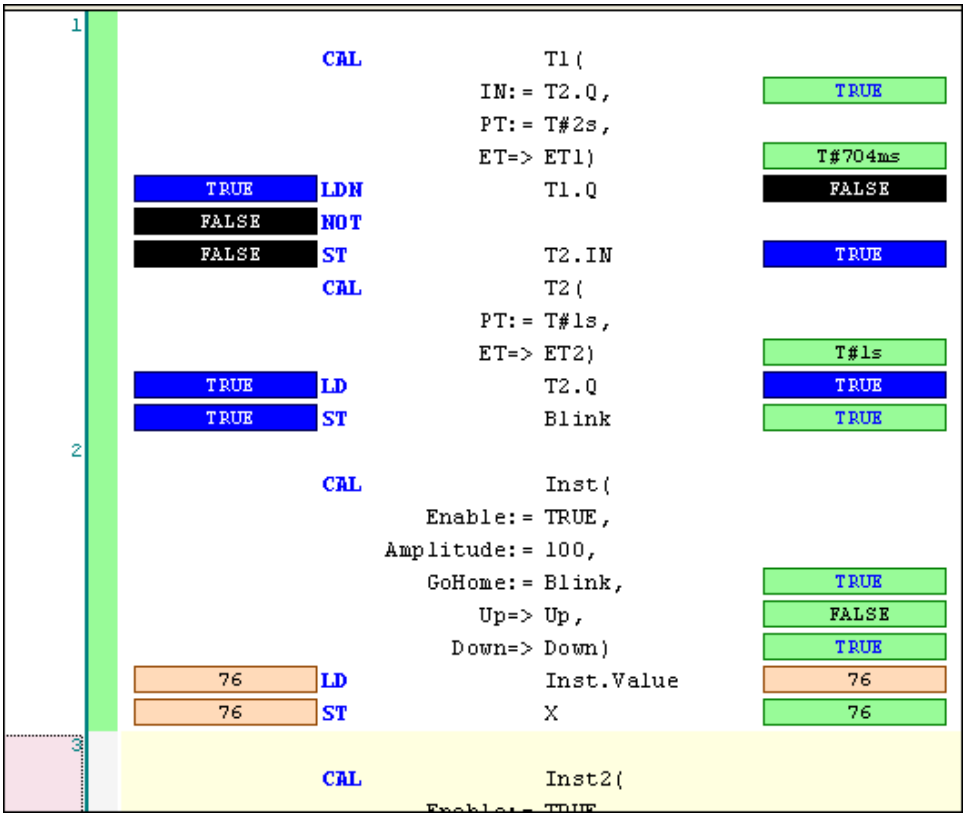


Abb. 200 Beispiel: Ablaufkontrolle im AWL-Editor

Man beachte die Möglichkeit, auch im Ablaufkontroll-Modus Werte zu schreiben. Forcen ist nicht möglich! Durch einen Doppeklicke auf das Werteanzeigefeld wird der Dialog Wert vorbereiten geöffnet, wo der gewünschte Wert eingegeben werden kann.

12.22.24 Darstellung

In diesem Untermenü (Kategorie Online) kann eingestellt werden, welches Format für die Darstellung der Werte (monitoring) im Online-Modus verwendet werden soll. Führen Sie dazu einen Mausklick auf die gewünschte Option aus, die daraufhin mit einem Häkchen gekennzeichnet wird.

- Binär
- Dezimal
- Hexadezimal

12.22.25 Verbinden mit Gerät

Dieser Befehl (Kategorie Online) kann verwendet werden, um eine Verbindung mit dem gerade im Gerätebaum ausgewählten Gerät herzustellen. Voraussetzung sind korrekte Kommunikationseinstellungen für dieses Gerät.

12.22.26 Trennen von Gerät

Dieser Befehl (Kategorie Online) kann verwendet werden, um die aktuell bestehende Verbindung zum im Gerätebaum ausgewählten Gerät zu beenden.

12.22.27 Reset Ursprung Gerät

Dieser Befehl (Kategorie Online) ist im Kontextmenü im Offline- und Online-Status verfügbar, wenn ein programmierbares Gerät im Gerätebaum ausgewählt ist. Er setzt das Gerät insofern in den Ursprungszustand (Lieferzustand) zurück, als alle Applikationen, Bootapplikationen und remanente Variablen auf dem Gerät gelöscht werden.

12.23 Einstellungen

Die Befehlskategorie 'Einstellungen' enthält Befehle für das Anpassen der Benutzeroberfläche und das Setzen verschiedener »PLC Designer«- und Projektoptionen.

12.23.1 Einstellungen

Befehle der Kategorie Einstellungen

Die 'Frame'-Komponente stellt Befehle für die Befehlskategorie 'Einstellungen' Befehle bereit, mit denen Dialoge zur Anpassung der Benutzeroberfläche und zum Setzen von Standardeinstellungen für verschiedene Komponenten (z.B. Editoren) geöffnet werden können.

Standardmäßig sind diese Befehle im 'Tools'-Menü enthalten. Falls nötig, kann aber auch die Definition dieses Menüs über den Anpassen-Dialog bearbeitet werden.

Die Befehle:

- Optionen
- Anpassen

Optionen-Dialog

Der Optionen-Dialog, durch das Frame Plug-In bereitgestellt, enthält Unterdialoge für die Konfiguration des Verhaltens und Aussehens der »PLC Designer« Benutzeroberfläche. Die Standardeinstellungen werden durch das aktuelle Profil vorgegeben.

Der Dialog wird mittels Befehl '**Optionen**' geöffnet (Befehlskategorie 'Einstellungen').

Die aktuellen Einstellungen im Optionen-Dialog werden auf dem lokalen System als Standardeinstellungen gespeichert und überschreiben die durch eine Neuinstallation von »PLC Designer« gelieferten Einstellungen.

Die Standard-Kategorien der Optionen:

- AS-Editor
- AS
- Bibliotheken
- CFC Editor
- »PLC Designer« 2.3 converter
- Deklarationseditor
- FUP, KOP und AWL
- Features
- Geräteeditor
- Intelligentes Kodieren
- Internationale Einstellungen
- Laden und speichern
- Quellcodeverwaltung
- Syntaxhervorhebung
- Texteditor
- Visualisierungen

Optionen, Laden und speichern

Dieser Unterdialog des Optionen-Dialogs enthält Einstellungen bezüglich des Verhaltens von »PLC Designer« beim Laden und Speichern eines Projekts. Die Optionen können durch Setzen eines Hakens im Kontrollkästchen aktiviert werden bzw. bestimmte Werte können eingegeben werden:

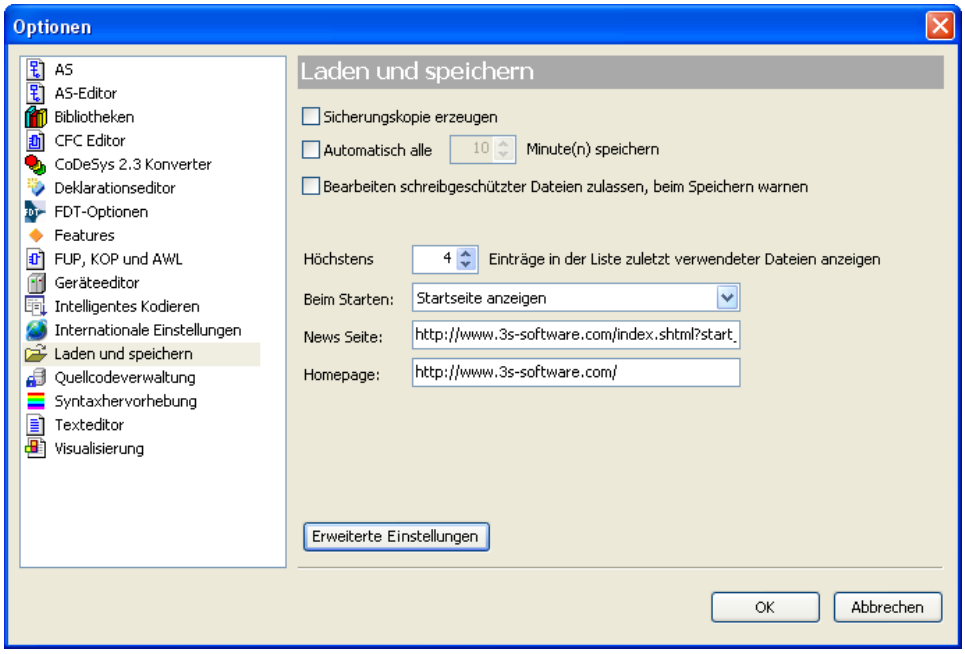


Abb. 201 Optionen-Dialog, Kategorie Laden und speichern

Sicherungskopie erzeugen: Wenn die Option aktiviert ist wird bei jedem Speichern das Projekt nicht nur in die Datei *<Projektname>.project* gespeichert, sondern zusätzlich auch in eine Datei *<Projektname>.backup* kopiert. Wenn nötig, können Sie dies backup-Datei umbenennen und wieder im Programmiersystem öffnen.

Automatisch alle ... Minute(n) speichern: Wenn die Option aktiviert ist, wird das Projekt automatisch im angegebenen Zeitintervall in eine Datei *<Projektname>.autosave* im Projektverzeichnis gespeichert, die dann nach einem nicht-regulären Schließen des Programmiersystems wieder geladen werden kann.

Bei regulärem Schließen oder Speichern des Projekts wird die zugehörige autosave-Datei gelöscht, bei nicht-regulärem Abbruch wird sie behalten. Wenn Sie ein Projekt öffnen, für das eine zugehörige autosave-Datei gefunden wird, wird der **Auto Save Backup**-Dialog geöffnet, in dem Sie angeben, ob Sie die autosave-Datei oder die zuletzt vom Benutzer gespeicherte Version des Projekts öffnen wollen:

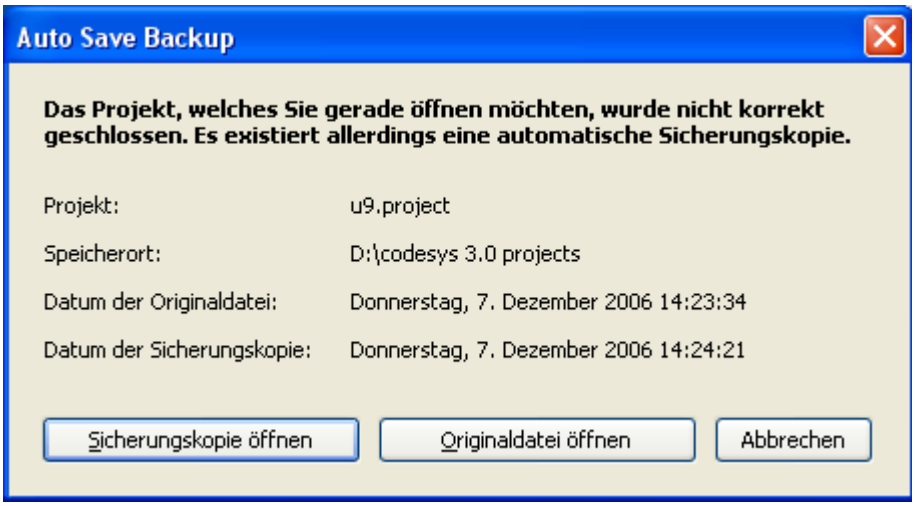


Abb. 202 Auto Save Backup Dialog

Sicherungskopie öffnen: Die autosave-Datei <*Projektname*>.*autosave* wird geöffnet und als geändert markiert ("*" hinter Dateiname in der Titelleiste)

Originaldatei öffnen: Die zuletzt durch den Anwender gespeicherte Version des Projekts wird geöffnet.

Bearbeiten schreibgeschützter Dateien zulassen, beim Speichern warnen: Wenn diese Option aktiviert ist, können Sie auch ein Projekt bearbeiten, für das nur Leserecht besteht. Allerdings wird eine Warnung ausgegeben, wenn Sie das Projekt speichern wollen.

Höchstens ... Einträge in der Liste zuletzt verwendeter Dateien anzeigen: Geben Sie hier an, wie viele der zuletzt bearbeiteten Projekt in der Liste "Zuletzt verwendete Dateien" (standardmäßig über Datei-Menü erreichbar) angezeigt werden sollen.

Beim Starten: Stellen Sie hier mit einer der folgenden Optionen ein, was beim Start von »PLC Designer« automatisch geschehen soll:

- Das zuletzt verwendete Projekt öffnen: Das Projekt, das zuletzt geöffnet war, wird automatisch wieder geladen.

- Den Dialog "Projekt öffnen" anzeigen: Der Dialog wird geöffnet, entspricht der Verwendung von Befehl 'Projekt Öffnen'.

- Den Dialog "Neues Projekt" anzeigen: Der Dialog wird geöffnet, entspricht der Verwendung von Befehl 'Neues Projekt'.

- Eine leere Umgebung anzeigen: Das Programmiersystem startet, ohne ein Projekt neu anzulegen oder ein bestehendes Projekt oder Dialoge zu öffnen.

- Startseite anzeigen: Die Ansicht 'Startseite' wird geöffnet. Entspricht Befehl Startseite.

News Seite: URL der Seite, die im 'News' Bereich der Startseite erscheint

Homepage: URL der Seite, die mit Befehl 'Homepage' (Startseite) oder '3S Homepage' (Hilfe Menü)geöffnet wird.

Erweiterte Einstellungen...: Schaltfläche zum Öffnen des gleichnamigen Dialogs:

Projektkomprimierung: Sehen Sie im Dialog weitere Informationen zu den drei möglichen Komprimierungsstufen. Standardmäßig ist die geringste Komprimierung eingestellt.

Ladeverhalten: Wenn diese Option aktiviert ist (Standardeinstellung), erfolgt das Laden von Bibliotheken und Übersetzungsinformation im Hintergrund, während man bereits mit dem Bearbeiten des Projekts beginnen kann.

Optionen, Texteditor

Dieser Unterdialog des Optionen-Dialogs enthält Einstellungen für das Arbeiten in einem Texteditor:

1. Editieren: Definition von Rückgängig-Schritten, Tabulator, Einrücken, Umbruch etc.

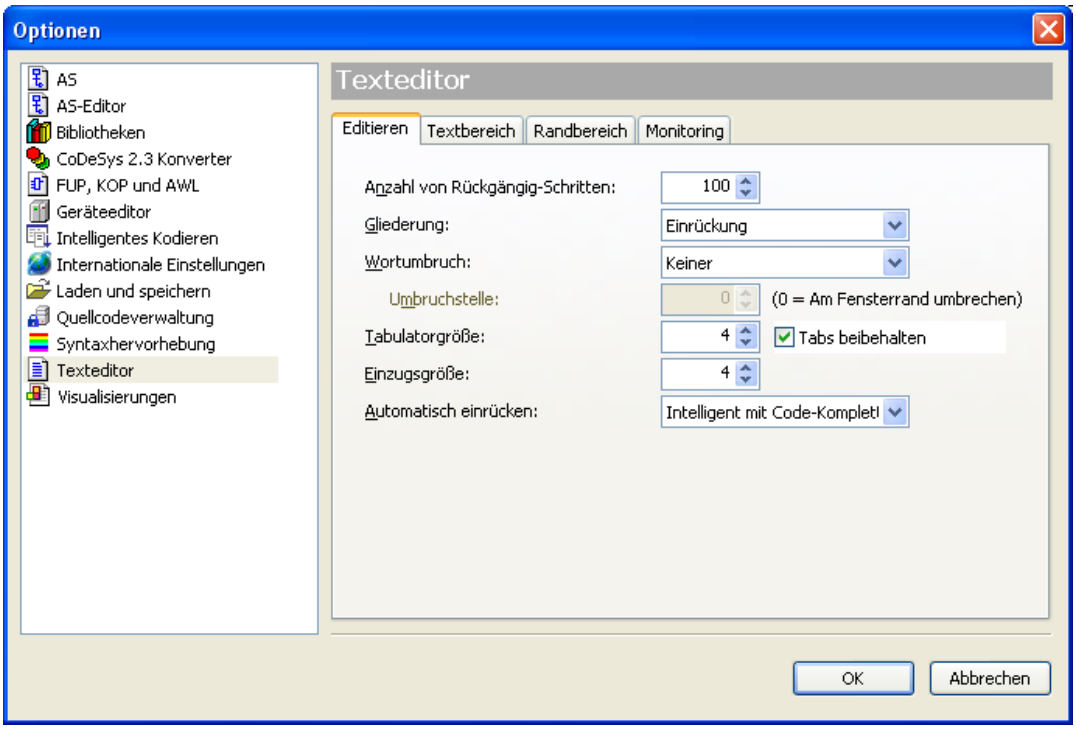
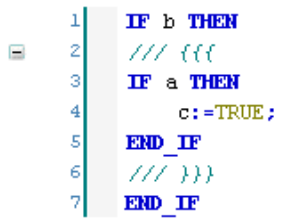
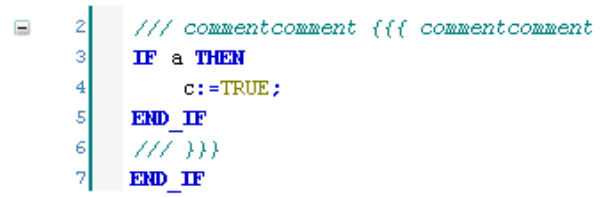


Abb. 203 Optionen-Dialog, Kategorie Texteditor, Editieren

Anzahl von Rückgängig-Schritten:	Die angegebene Zahl von Bearbeitungsschritten wird gespeichert und kann mit Hilfe des "Rückgängig"-Befehls rückgängig gemacht werden.
Gliederung:	Die gewählte Option legt fest, ob der Code durch Einrückungen strukturiert werden soll. Wenn ja, dann können eingerückte Abschnitte bzw. Abschnitte, die durch einen speziellen Kommentar gekennzeichnet werden, ausgeblendet oder eingeblendet werden, und zwar mit Hilfe von Plus- und Minuszeichen vor der ersten Zeile des jeweiligen Abschnitts. Folgende Optionen sind möglich: **Keine**: Keine Einrückung. **Einrückung**: Alle Zeilen, die gegenüber der vorausgehenden eingerückt sind, werden in einer "Einrückungseinheit" zusammengefaßt, der die vorausgehenden Zeile vorsteht. Beispiele: Die Zeilen zwischen VAR und END VAR bzw. zwischen IF und END_IF sind eingerückt; somit werden sie in eine Einrückungseinheit gefaßt, die im geöffneten Zustand mit einem Minuszeichen und im geschlossenen Zustand mit einem Pluszeichen vor "VAR" gekennzeichnet ist: Geöffnete und geschlossene Einrückungseinheiten: fold opened fold closed fold opened fold closed **Explizit**: Wenn diese Option aktiviert ist, muss der Code-Abschnitt, der in einer Einrückungseinheit zusammengefaßt werden soll, explizit durch Kommentare gekennzeichnet werden: Vor dem Abschnitt muss ein Kommentar stehen, der drei öffnende geschweifte Klammern "{{{" enthält, nach dem Abschnitt muss ein Kommentar folgen, der drei schließende geschweifte Klammern "{{{" enthält. Die Kommentare können zusätzlichen Text enthalten.

	Beispiele für die Defintion eines eingerückten Abschnitts   *Abb. 204 Explizit durch spezielle Kommentare definierte Einrückungsabschnitte:*
Wortumbruch:	**Keiner**: Die Zeilen werden endlos weitergeführt. **Weich**: Die Zeilen werden am Rand des Editorfensters umgebrochen, wenn bei **Umbruchsstelle** "0" eingetragen ist. **Hart**: Die Zeilen werden nach der bei **Umbruchsstelle.** angegebenen Anzahl von Zeichen umgebrochen.
Tabulatorgröße:	Tabulatorgröße in Anzahl Zeichen.
Einzugsgröße:	Wenn die Option Automatisch Einrücken (s.u.) aktiviert ist, wird um die hier angegeben Anzahl Zeichen nach rechts eingerückt, d.h. es werden entsprechend viele Leerzeichen am Beginn der Zeile eingefügt.
Automatisch einrücken:	**Nie**: Während des Editierens findet kein automatisches Einrücken statt. Eingegebene Zeichenfolgen beginnen immer am linken Rand des Textbereichs **Block:** **Intelligent**: Die Zeilen, die einer Zeile folgen, die ein Schlüsselwort enthält (z.B. VAR oder IF) werden automatisch um die oben angegebene Einzugsgröße eingerückt. **Intelligent mit Code-Komplettierung**: Die Zeilen, die einer Zeile folgen, die ein Schlüsselwort enthält (z.B. VAR oder IF) werden automatisch um die oben angegebene Einzugsgröße eingerückt und zusätzlich wird das entsprechende abschließende Schlüsselwort eingefügt (z.B. END_VAR oder END_IF).
Tabs beibehalten	Wenn diese Option aktiviert ist, wird ein Leerraum in der Zeile, der mit der <Tabulator>-Taste gemäß der definierten Tabulatorgröße (s.o.) eingefügt wurde, hinterher nicht in einzelne Leerzeichen aufgelöst. Das bedeutet, dass wenn die Tabulatorgröße verändert wird, auch die mit <Tabulator> erzeugten Leerräume angepaßt werden.

2. Textbereich: Farb- und Font-Definitionen für den Textbereich.

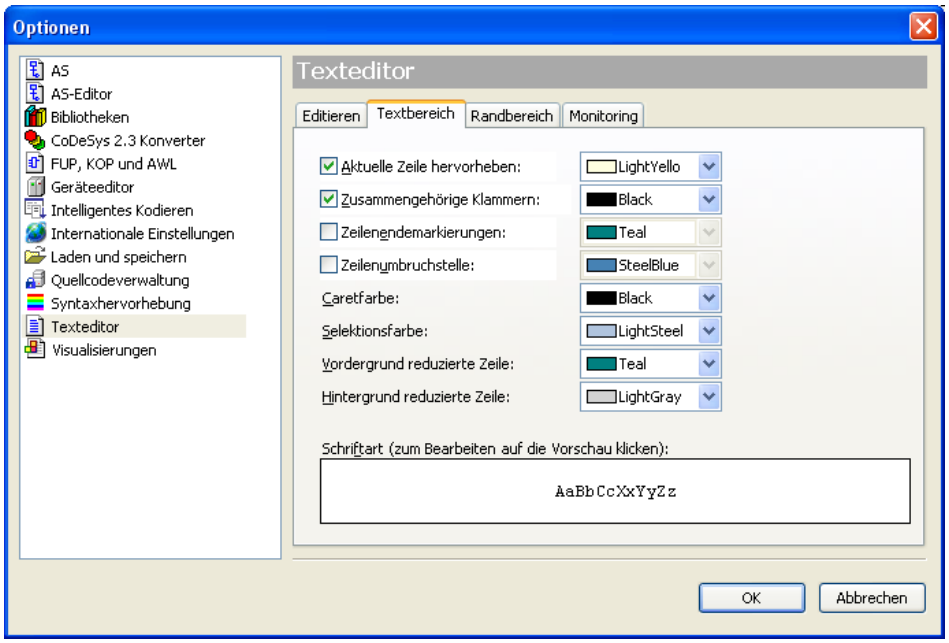


Abb. 205 Optionen-Dialog, Kategorie Texteditor, Textbereich

Die folgenden Optionen können aktiviert werden, um beim Editieren optische Unterstützung zu bekommen :

Aktuelle Zeile hervorheben:	Die Zeile in der der Cursor steht, wird mit der eingestellten Farbe hinterlegt.
Zusammengehörige Klammern:	Wenn der Cursor vor oder nach einer Klammer "(" oder ")" in einer Code-Zeile steht, wird die zugehörige schließende bzw. öffnende Klammer durch einen Rahmen in der eingestellten Farbe markiert. Ebenso wird, wenn der Cursor in der ersten oder letzten Zeile einer Klammerung die entsprechende End- oder Anfangszeile dieses Bereichs durch eine rechteckige Umrahmung in dieser Farbe angezeigt.
Zeilenendmarkierungen:	Das Ende jeder Editorzeile wird durch einen kleinen Querstrich in der eingestellten Farbe hinter dem letzten Zeichen (auch Leerzeichen) der Zeile markiert.
Zeilenumbruchstelle:	Wenn ein weicher oder harter Zeilenumbruch aktiviert ist (s.o., Dialog 'Editieren'), wird die definierte Zeilenumbruchstelle durch eine senkrechte Linie in der gewählten Farbe angezeigt.
Caretfarbe:	Farbe des Cursorzeichens.
Selektionsfarbe:	Der gerade selektierte Textbereich wird mit der gewählten Farbe hinterlegt.
Vordergrund reduzierte Zeile:	Die Kopfzeile eines geschlossenen, eingerückten Abschnitts im Code wird in der gewählten Farbe dargestellt.
Hintergrund reduzierte Zeile:	Die Kopfzeile eines geschlossenen, eingerückten Abschnitts im Code wird mit der gewählten Farbe hinterlegt.
Schriftart:	Die hier eingestellte Schriftart wird im Texteditor verwendet. Durch einen Mausklick auf das Beispielfeld wird der Standarddialog zum Konfigurieren der Schriftart geöffnet.

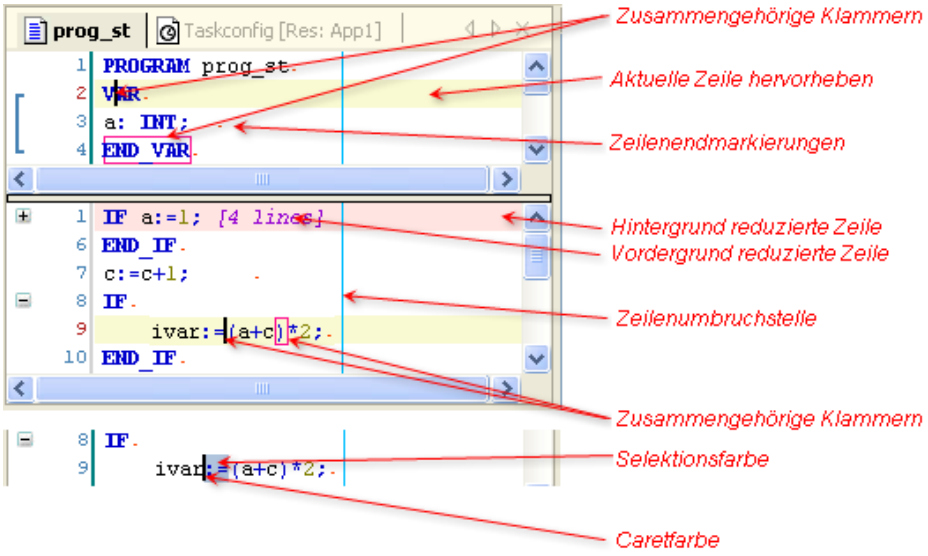


Abb. 206 Beispiele für die Einstellungen im Textbereich

3. Randbereich: Definitionen von Farben, Font, Mausaktionen im Randbereich des Editors, links vom Textbereich.

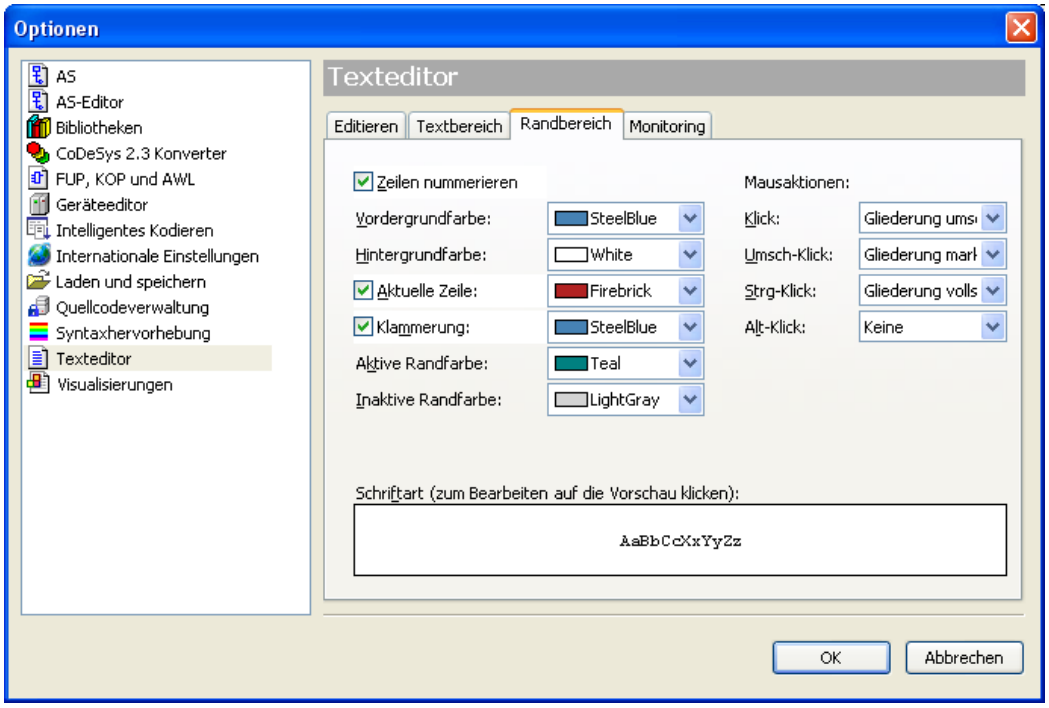


Abb. 207 Optionen-Dialog, Kategorie Texteditor, Randbereich

Die folgenden Einstellungen und Optionen betreffen den linken Randbereich des Texteditor-Fensters, der durch eine senkrechte Linie vom Eingabebereich abgetrennt ist:

Zeilen nummerieren:	Zeilennummern werden im Deklarations- und Implementationsteil des Texteditors angezeigt, jeweils mit 1 in der ersten Zeile beginnend.
Vordergrundfarbe:	Farbe der Zeilennummern.
Hintergrundfarbe:	HIntergrundfarbe des Randbereichs.
Aktuelle Zeile:	Farbe der Zeilennummer der Zeile, in der gerade der Cursor steht.
Klammerung:	Eine "Klammerung" enthält die Zeilen zwischen den Schlüsselwörtern, die ein Konstrukt öffnen und abschließen, z.B. zwischen IF und END_IF. Die Klammerungs-Funktion kann über die Option 'Zusammengehörige Klammern' im Dialog 'Textbereich' aktiviert werden. Wenn der Cursor dann vor, nach oder in einem der korrespondierenden Schlüsselwörter steht, wird der Klammerungsbereich durch eine Klammerlinie im Randbereich angezeigt.
Aktive Randfarbe:	Farbe der senkrechten Trennlinie zwischen Randbereich und Eingabebereich des gerade aktiven Teils des Texteditor-Fensters.
Inaktive Randfarbe:	Farbe der senkrechten Trennlinie zwischen Randbereich und Eingabebereich des gerade nicht aktiven Teils des Texteditor-Fensters.
Mausaktionen:	Eine der folgenden Aktionen kann jeder der angegebenen Mausaktionen bzw. Maus-Tastenkombinationen (Klick, <Umsch>-Klick, <Strg>-Klick, <Alt>-Klick) zugeordnet werden. Sie werden ausgeführt, wenn die Mausaktion auf das Plus- bzw. Minuszeichen vor der Kopfzeile eines geklammerten Bereichs ausgeführt wird: **None**: Keine Aktion. **Gliederung markieren**: Alle Zeilen des geklammerten Bereichs werden ausgewählt. **Gliederung umschalten**: Der geklammerte Bereich, bzw. wenn geschachtelte Klammerungen vorliegen, die erste Ebene des geklammerten Bereichs, wird geöffnet bzw. geschlossen. **Gliederung vollständig**: Alle Ebenen eines geschachtelt geklammerten Bereichs werden geöffnet bzw. geschlossen.

4. Monitoring: Aktivieren/Deaktivieren des Inline-Monitorings, Darstellung der Monitoring-Felder.

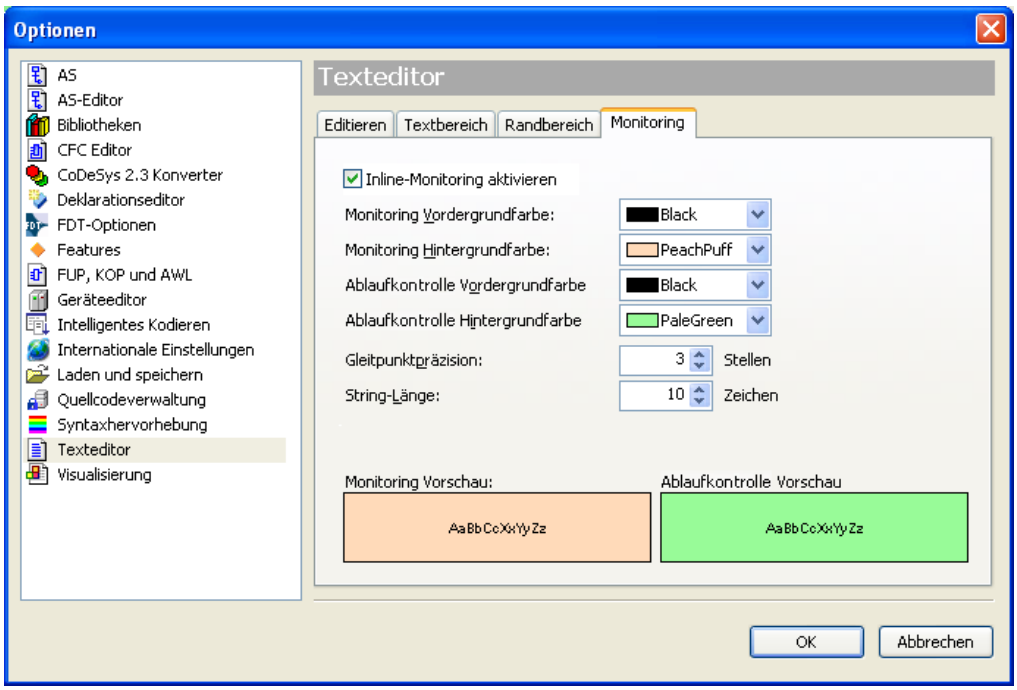


Abb. 208 Optionen-Dialog, Kategorie Monitoring

Inline-Monitoring aktivieren:	Die Anzeige der Monitoring-Felder hinter Variablen im Online-Modus eines Texteditors kann aktiviert und deaktiviert werden.
Monitoring Vordergrundfarbe:	Der Wert im Monitoring-Feld wird in der ausgewählten Farbe dargestellt.
Monitoring Hintergrundfarbe:	Der Hintergrund des Monitoring-Felds wird in der ausgewählten Farbe dargestellt.
Ablaufkontrolle Vordergrundfarbe:	Der Wert in den Monitoring-Feldern an den Ablaufkontrollpositionen wird in der ausgewählten Farbe dargestellt.
Ablaufkontrolle Hintergrundfarbe:	Der Wert in den Monitoring-Feldern an den Ablaufkontrollpositionen wird in der ausgewählten Farbe dargestellt.
Gleitpunktpräzision:	Kommazahlen im Monitoring-Feld werden mit der hier eingestellten Anzahl von Stellen hinter dem Komma dargestellt.
String-Länge:	String-Variablenwerte werden maximal in der hier angegebenen Länge (Anzahl von Zeichen) im Moniitoring-Feld dargestellt.
Vorschau:	Vorschau mit den gerade eingestellten Werten für ein Monitoring-Feld.

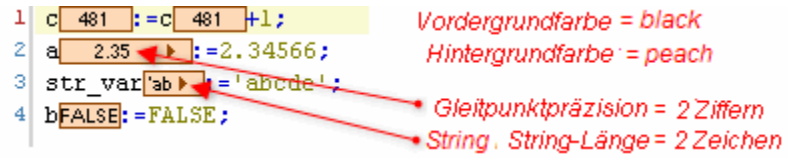


Abb. 209 Beispiel Inline-Monitoring im ST-Editor

Optionen, Internationale Einstellungen

Dieser Unterdialog des Optionen-Dialogs enthält die folgenden Einstellungen bezüglich der Sprache in der Benutzeroberfläche und Online Hilfe.

Sprache in der Benutzeroberfläche:

Dieselbe Sprache wie Microsoft Windows verwenden: Wenn dies Option aktiviert ist, wird diejenige Sprache verwendet, die gerade Microsoft Windows auf dem lokalen System eingestellt ist.

Spezielle Sprache: Wenn diese Option aktiviert ist, wird die hier gewählte Sprache verwendet.

Hinweis: Eine hier vorgenommene Änderung der Spracheinstellung wird erst beim nächsten Neustart von »PLC Designer« wirksam. Komponenten, die möglicherweise nicht in der ausgewählten Sprache verfügbar sind, werden in der Default-Sprache (typischerweise Englisch) erscheinen.

Online Hilfe Sprache:

Dieselbe Sprache wie die Benutzeroberfläche verwenden: Diese Option ist standardmäßig gesetzt. Die Online Hilfe wird in der Sprache geöffnet, die für die Benutzeroberfläche gesetzt ist. Wenn in dieser Sprache keine Hilfedateien vorliegen, wird Englisch verwendet.

Spezielle Sprache: Wenn diese Option aktiviert ist, wird die hier gewählte Sprache verwendet.

Optionen, Intelligentes Kodieren

Dieser Unterdialog des Optionen-Dialogs enthält Einstellungen, mit denen die Eingabe von Code erleichtert werden kann.

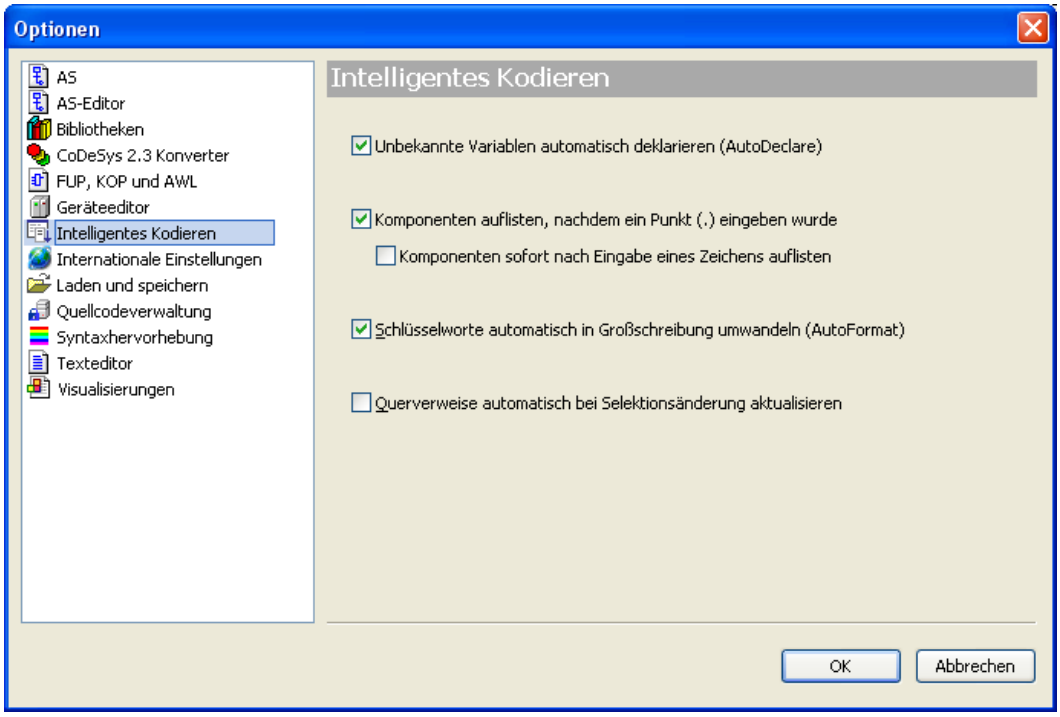


Abb. 210 Optionen-Dialog, Kategorie 'Intelligentes Kodieren'

Unbekannte Variablen automatisch deklarieren (AutoDeclare): Wenn diese Option aktiviert ist, wird automatisch der Dialog Variable deklarieren geöffnet, sobald ein noch nicht deklarierter Bezeichner in einem Spracheditor eingegeben wird und die Eingabezeile verlassen wird.

Komponenten auflisten, nachdem ein Punkt (.) eingegeben wurde: Mit dieser Option wird die "Komponenten Auflisten"-Funktionalität aktiviert. Das bedeutet: Wenn in einem Editor an einer Stelle, an der ein Bezeichner erwartet wird, ein Punkt (".") eingegeben wird, erscheint eine Auswahlliste der an dieser Stelle möglichen Eingaben.

Komponenten sofort nach Eingabe eines Zeichens auflisten: Wenn diese Option aktiviert ist, erscheint nach Eingabe einer beliebigen Zeichenfolge in einem Editor eine Auswahlliste der verfügbaren Bezeichner und Operatoren. Dies ist ebenfalls eine Form von "Komponenten Auflisten"-Funktionalität. Der erste Eintrag in der Auswahlliste, der mit der gleichen Zeichenfolge beginnt wie die, die eingegeben wurde, ist automatisch bereits als ausgewählt markiert und kann mit <Eingabe> an die Cursor-Position übernommen werden. Ansonsten kann auch ein anderer Eintrag aus der Liste gewählt werden.

Schlüsselworte automatisch in Großschreibung umwandeln (Autoformat): Wenn diese Option aktiviert ist, werden alle Schlüsselwörter automatisch in Großbuchstaben geschrieben. Beispiel: Wenn "bVar:bool;" eingegeben wird, wird es zu "iVar:BOOL;" umgewandelt.

Querverweise automatisch bei Selektionsänderung aktualisieren:Wenn diese Option aktiviert ist, zeigt die Querverweisliste immer automatisch die Liste der Referenzen der Variablen, die gerade im Editor den Fokus hat.

Unterschiede zu »PLC Designer« V2.3 bezüglich der Querverweisliste:

- Die Querverweisliste kann in einem Fenster (Ansicht) neben dem Editorfenster angezeigt werden.
- Es ist nicht erforderlich, das Projekt zu kompilieren, um die Querverweise angezeigt zu bekommen.
- Auch Komponentenzugriffe, Enumerationskonstanten und benutzerdefinierte Datentypen werden angezeigt.
- Eine dynamische Anzeige der Querverweise für jeweils die gerade fokusierte Variable kann aktiviert werden.

Optionen, Syntaxhervorhebung

Dieser Unterdialog des Optionen-Dialogs enthält Farb- und Schriftarteinstellungen für die verschiedenen Textelemente in einem Editor (z.B. Operanden, Pragmas, Kommentare etc.)

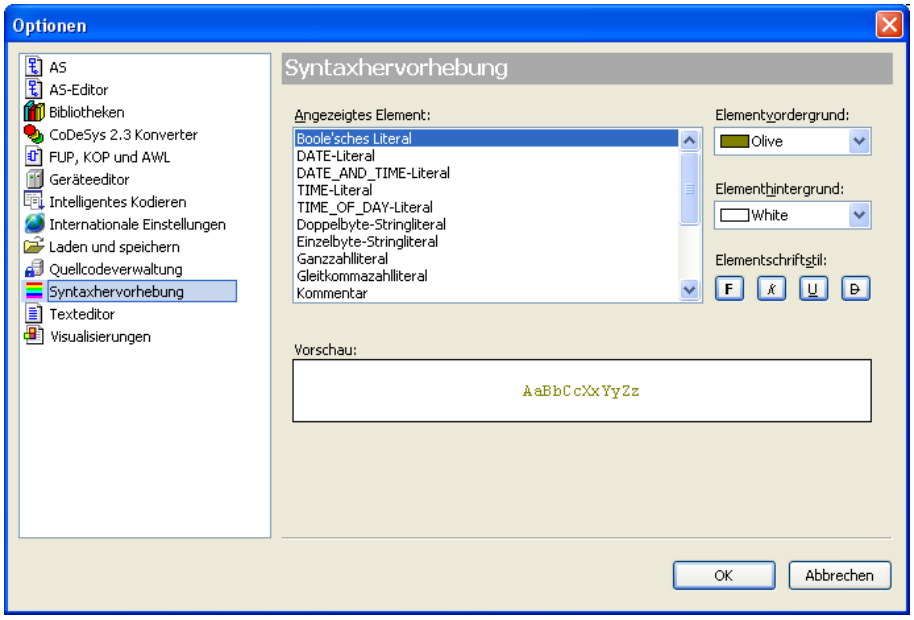


Abb. 211 Optionen-Dialog, Kategorie Syntaxhervorhebung

Für die in der Liste **Angezeigtes Element** angegebenen Textelemente kann jeweils die Farbe und die Schriftart eingestellt werden, in denen es im Texteditor dargestellt werden soll. Die Farbe kann für den **Elementvordergrund** und den **Elementhintergrund** durch mit Hilfe einer Auswahlliste eingestellt werden. Der **Elementschriftstil** wird durch Anklicken der gewünschten Symbole für "fett" (**B**), "kursiv" (*I*), "unterstrichen" (U) und "durchgestrichen"(S) definiert.

Eine Beispiel mit den aktuellen Einstellungen gibt es im **Vorschau**-Fenster.

Optionen, Bibliotheken

In diesem Unterdialog des Optionen-Dialogs werden die "**Mappings**" von **Bibliotheksreferenzen** verwaltet, die bei der Konvertierung eines "alten" Projekts angewendet werden. Solange hier für eine bestimmte Bibliothek noch kein "Mapping" gespeichert ist, muss es der Benutzer jedes Mal, wenn er ein altes Projekt öffnet, in das diese Bibliothek eingebunden ist, neu definieren.

Siehe Datei öffnen Informationen zum Öffnen und Konvertieren von Projekten ins aktuelle Format.

Siehe Bibliotheksverwatlung für Informationen zur Bibliotheksverwaltung.

Ein "Mapping" definiert, wie eine Bibliotheksreferenz nach der Konvertierung des Projekts ins aktuelle Format aussehen soll. Es gibt drei Möglichkeiten:

- Die Referenz wird beibehalten, d.h. die Bibliothek wird ebenfalls ins aktuelle Format (*.library) konvertiert und im lokalen Bibliotheks-Repository installiert.
- Die Referenz wird durch eine andere ersetzt, d.h. eine der installierten Bibliotheken ersetzt die bisher eingebundene.
- Die Referenz wird gelöscht, d.h. die Bibliothek wird im konvertierten Projekt nicht mehr eingebunden sein.

Wenn das "Mapping" gespeichert wird, wird es hier im Optionen-Dialog, Kategorie 'Bibliotheken' hinzugefügt:

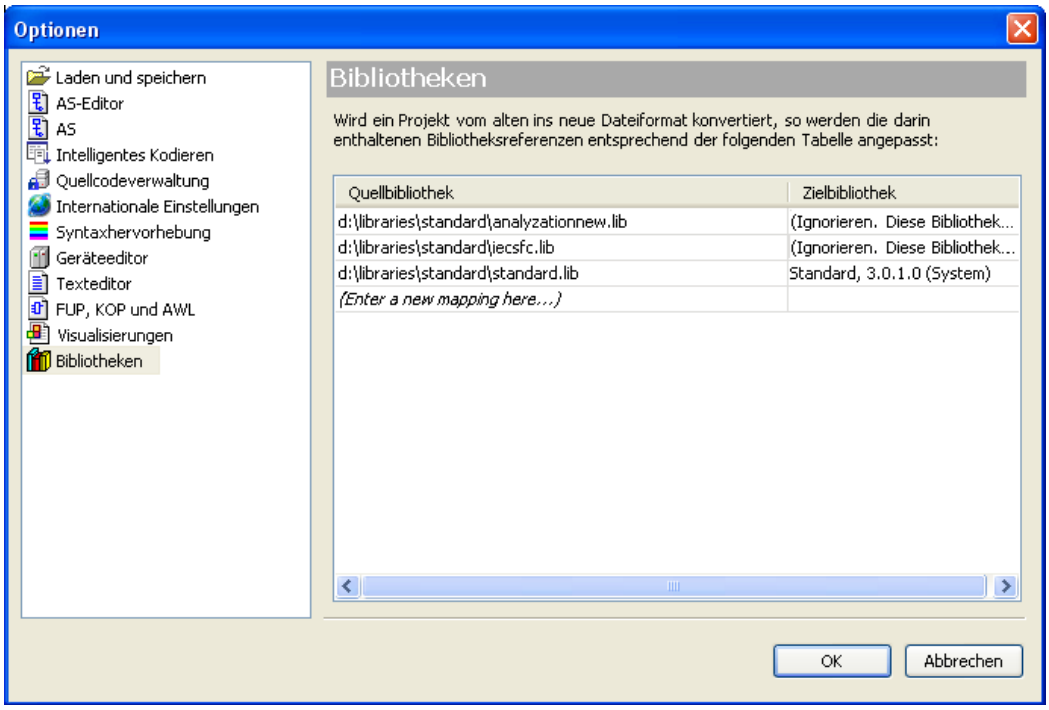


Abb. 212 Optionen-Dialog, Kategorie Bibliotheken

Alle Mappings, die hier aufgelistet sind, werden bei der nächsten Konvertierung eines alten Projekts auf dessen Bibliotheksreferenzen angewendet. Somit muss die Mapping-Definition nicht wiederholt werden, wenn dieselbe Bibliothek wieder in einem zu konvertierenden Projekt eingebunden ist.

Jedes "Mapping" ist in einer Zeile dargestellt:

Quellbibliothek: Pfad der Bibliothek, die im Projekt vor der Konvertierung eingebunden ist.

Zielbibliothek: Name und Speicherort der Bibliothek, die nach der Konvertierung im Projekt eingebunden sein soll. Wenn die Referenz entfernt werden soll, wird dies durch den Text "(Ignorieren. Diese Bibliotheksreferenz wird nicht konvertiert.)"angezeigt.

Die Mappings-Liste kann bearbeitet werden: Dazu führen Sie einen Doppelklick auf das jeweilige Feld aus oder selektieren ein Feld und drücken <Eingabe>.

Wenn ein Quellbibliotheks-Eintrag editiert wird, öffnet Schaltfläche ... den Standard-Browse-Dialog, in dem nach einer Bibliothek im "alten" Format gesucht werden kann.

Wenn ein Zielbibliotheks-Eintrag editiert wird, öffnet der Dialog 'Zielsystembibliothek setzen':

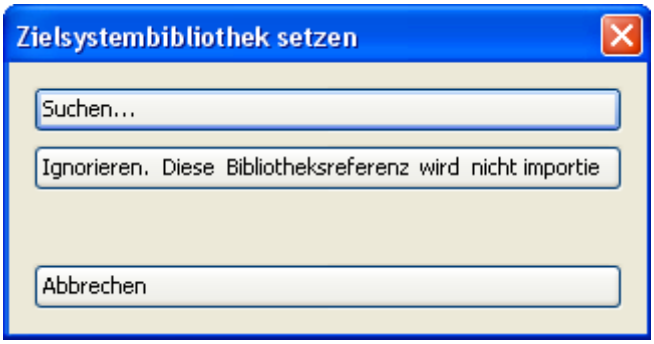


Abb. 213 Dialog 'Zielsystembibliothek setzen'

Um eine Bibliothek aktuellen Formats zu öffnen, kann über Schaltfläche **Suchen** der Dialog **Bibliothek auswählen** geöffnet werden, wo eine der im lokalen Bibliotheks-Repository installierten Bibliotheken ausgewählt werden kann. Es kann nach Firma und Kategorie sortiert werden, wie es vom Bibliotheks-Repository-Dialog her bekannt ist.

Um festzulegen, dass die vorliegende Quellbibliothek immer aus dem Projekt entfernt werden soll, wenn dieses konvertiert wird, muss Schaltfläche **Ignorieren. ...** verwendet werden.

Wenn der Liste ein **neuer Eintrag hinzugefügt** werden soll, kann dies auch direkt hier im Dialog erfolgen, indem die letzte Zeile editiert wird "*(Neues Mapping hier eingeben.)*".

12.23.2 Optionen,»PLC Designer«-Konverter

In diesem Unterdialog des Optionen-Dialogs ist das "Mapping" für die Konvertierung von »PLC Designer« V2.3 Projekten definiert. D.h. es wird festgelegt, wie die Komponenten eines Projekts beim Überführen in das aktuelle Projektformat gehandhabt werden.

Im Registerblatt 'Geräte' wird die Konvertierung der Geräte-Referenzen (welches Zielsystem wird verwendet) beschrieben:

Solange für ein Gerät kein "Mapping" hier im Dialog definiert ist, werden Sie immer aufgefordert, eines zu definieren, wenn ein altes Projekt geöffnet wird, das dieses Gerät verwendet.

Siehe Projekt öffnen für eine Beschreibung zum Öffnen und Konvertieren alter Projekte.

Siehe Geräte, Gerätebaum für eine Beschreibung zur Handhabung von Geräten im Projekt.

Eine "Geräte-Konvertierungs-Mapping" definiert, wie die Referenz auf ein Zielsystem nach der Konvertierung des Projekts aussehen soll. D.h. ein "Ziel-Gerät" (Gerät im neuen Projekt) wird einem "Quell-Gerät" (ursprüngliches Gerät im alten Projekt) zugeordnet. Es gibt zwei Möglichkeiten:

- Das Quell-Gerät wird im Gerätebaum des konvertierten Projekts durch das Ziel-Gerät ersetzt, das aktuell auf dem System verfügbar, d.h. installiert ist.
- Das Quell-Gerät wird nicht konvertiert (wird ignoriert), d.h. es wird nicht im Gerätebaum des neuen Projekts erscheinen. Dadurch werden auch applikationsspezifische Objekte wie z.B. die Taskkonfiguration nicht ins neue Projekt übernommen.

Ein Geräte-Mapping kann definiert und gespeichert werden ...

- ... während der Konvertierung eines alten Projekts, das das Gerät verwendet (siehe hierzu die Beschreibung des Befehls 'Projekt öffnen')
- ... oder es durch direktes Eintragen hier im Konverter-Optionen-Dialog im Registerblatt 'Geräte':

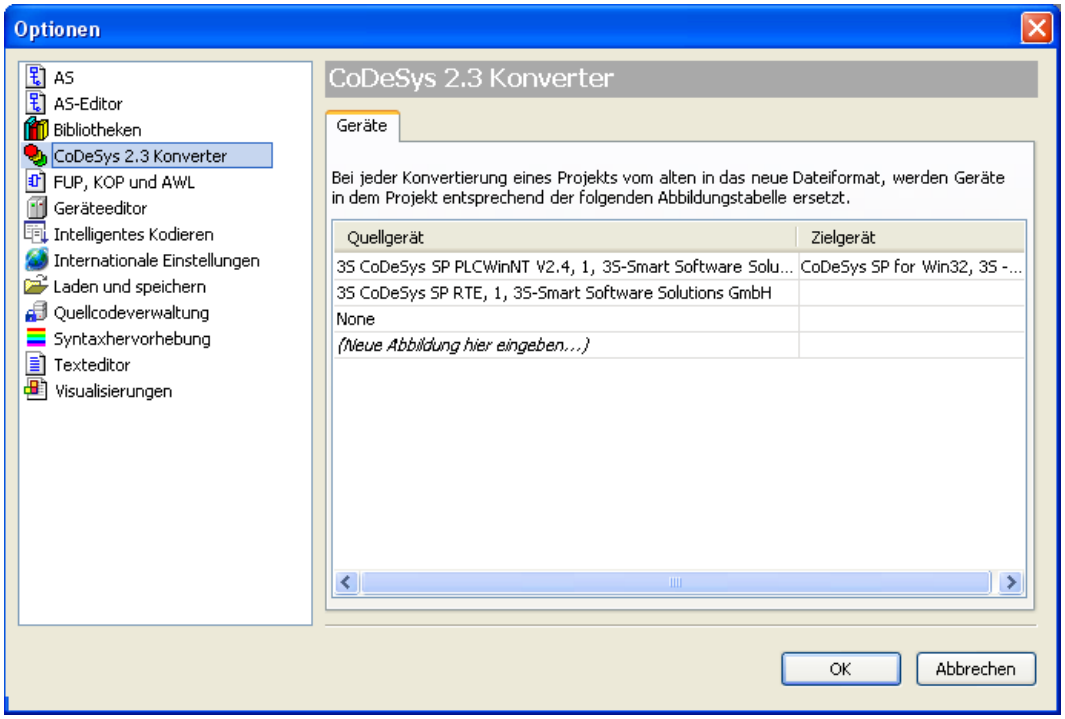


Abb. 214 Optionen-Dialog, Kategorie »PLC Designer« 2.3 Konverter

Um ein Feld in der Mapping-Tabelle zu bearbeiten, führen Sie einen Doppelklick auf das Feld aus, oder sie selektieren es und drücken die <Eingabe>-Taste. Weisen Sie jedem "Quell-Gerät" ein "Ziel-Gerät" zu:

Für Einträge in der Spalte 'Quellgerät' wird der Dialog 'Ursprüngliches Gerät auswählen' geöffnet, in dem alle Geräte aufgelistet sind, die vom Konverter gehandhabt werden können. Wählen Sie das gewünschte aus, um es im Feld einzufügen.

Für Einträge in der Spalte 'Quell-Gerät' wird der Dialog 'Zielsystem auswählen' geöffnet, in dem eines der aktuell installierten Geräte ausgewählt werden kann. Die Handhabung des Dialogs (Sortierung nach Hersteller und Kategorie) entspricht der des Device-Repository Dialogs. (Sehen Sie auch die Beschreibung zur Konvertierung beim Öffnen eines Projekts). Wenn 'Keines' aktiviert wird, wird das Quell-Gerät im neuen Projekt nicht verfügbar sein, was auch bedeutet, dass applikationsspezifische Objekte wie z.B. die Taskkonfiguration nicht übernommen werden!

Alle Geräte-Mappings, die im Optionen-Dialog aufgelistet sind, werden bei der nächsten Konvertierung eines alten Projektes angewendet. Somit muss die Mapping-Definition nicht jedesmal neu vorgenommen werden, wenn das gleiche Gerät in mehreren Projekten verwendet ist.

12.23.3 Optionen, CFC Editor

Dieser Unterdialog des Optionen-Dialogs erlaubt Einstellungen, die das Editieren im CFC (Continuous Function Chart) Editor betreffen.

Die Einstellungen werden sobald sie mit **OK** bestätigt wurden (was den Dialog schließt), unmittelbar in den gerade geöffneten Editor-Ansichten wirksam.

Einstellungen:

Automatisches Verknüpfen aktivieren: Wenn diese Option aktiviert ist, gilt folgendes: Wenn ein CFC Element irgendwo auf die Arbeitsfläche des Editors gezogen und eingefügt wird, werden unverknüpfte Pins, die sich "berühren", automatisch verbunden. Dies kann schnelleres Editieren unterstützen, achten Sie aber darauf, beim Verschieben von Elementen nicht zufällig unerwünschte Verknüpfungen zu erzeugen!

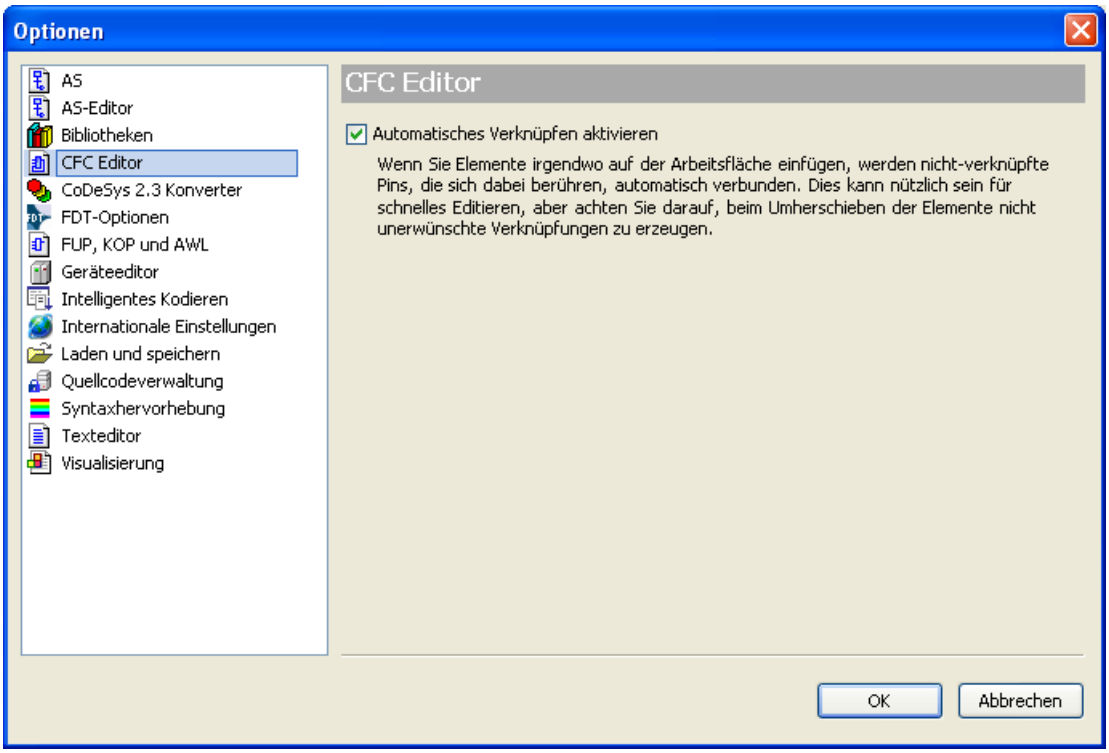


Abb. 215 CFC Editor Optionen

12.23.4 Optionen, Deklarationseditor

Dieser Teil des Optionen-Dialogs enthält Standardeinstellungen bezüglich der Anzeige des Deklarationseditors.

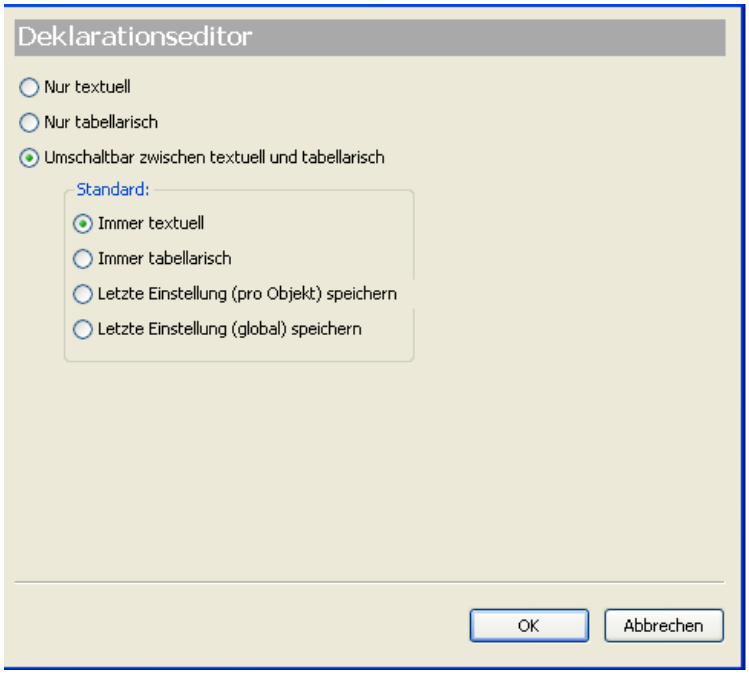


Abb. 216 Optionen Dialog, Deklarationseditor

Welche Ansicht des Deklarationseditors soll erscheinen, wenn ein Objekt zur Bearbeitung geöffnet wird:

Nur textuell

Nur tabellarisch

Umschaltbar zwischen textuell und tabellarisch: Wenn diese Einstellung aktiviert ist, gibt es im Deklarationseditor-Fenster zwei Schaltflächen, mit denen zwischen der textuellen und tabellarischen Form gewechselt werden kann. In diesem Fall definiert eine der folgenden Einstellungen, welche der beiden Ansichten **standardmäßig** zunächst beim Öffnen eines Objekts im Editor verwendet wird:

- **Immer textuell**
- **Immer tabellarisch**
- **Letzte Einstellung (pro Objekt) speichern**: Wenn ein Objekt erneut geöffnet wird, erscheint der Deklarationseditor in der gleichen Ansicht wie beim letzten Bearbeitungsvorgang.
- **Letzte Einstellung (global) speichern**: Wenn ein Objekt geöffnet wird, erscheint der Deklarationseditor in der gleichen Ansicht wie bei seiner letzten Verwendung, egal in welchem Objekt.

12.23.5 Optionen, FUP, KOP und AWL

In diesem Teil des Optionen-Dialogs wird der FUP/KOP/AWL-Editor konfiguriert:

Kategorie 'View':

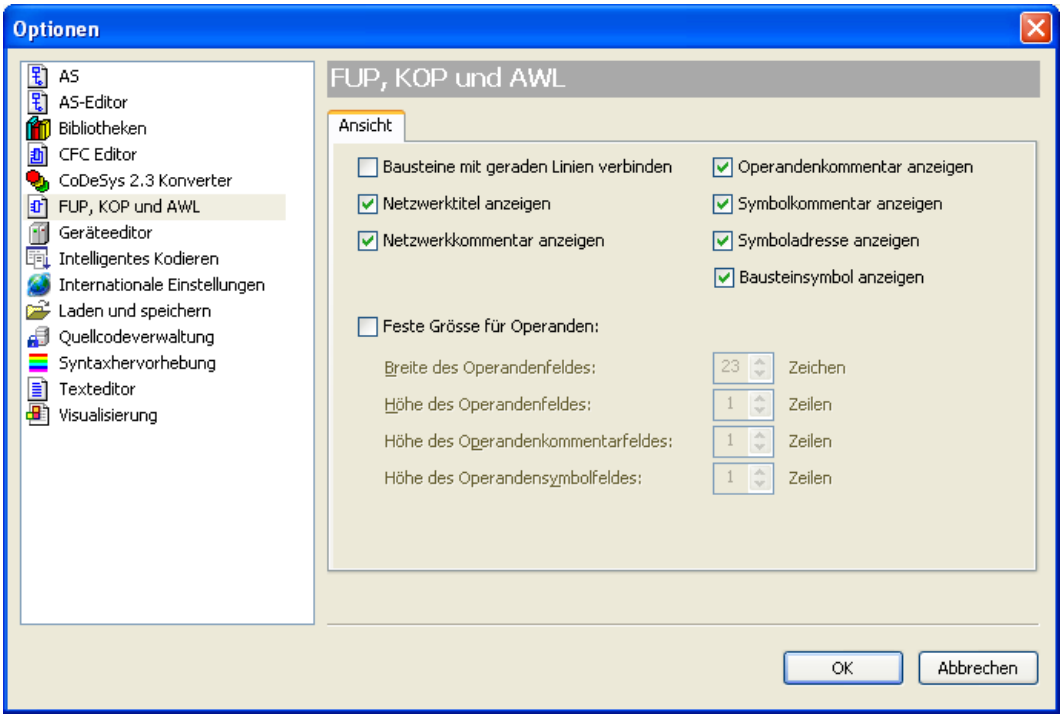
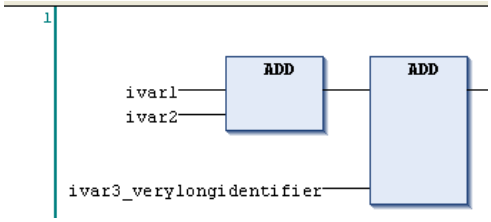
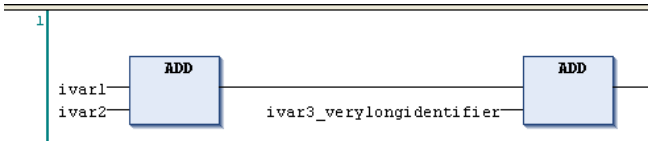
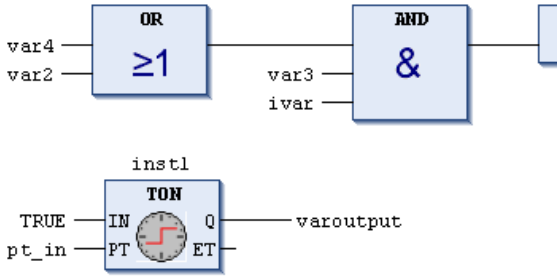


Abb. 217 Optionen für die Darstellung (View) des FUP/KOP/AWL Editors

Elemente mit geraden Linien verbinden	Wenn diese Option aktiviert ist, werden die Elemente in einem Netzwerk so angeordnet, dass die Linien zwischen ihnen eine feste, kurze Länge erhalten und somit die für die Darstellung der Netzwerke benötigte Breite soweit als möglich reduzieren. Dies kann bewirken, dass die Baustein-Boxen vertikal verlängert werden, um genügende Platz für alle Eingangs- und Ausgangselemente zu schaffen. Wenn die Option nicht aktiviert ist, behalten die Elemente ihre Standardgröße und die Verbindungslinien passen sich dem benötigten Raumbedarf an. Beispiel. Option aktiviert:  Option deaktiviert: 

Netzwerktitel anzeigen	Der Netzwerktitel - falls einer definiert ist - wird in der oberen linken Ecke des Netzwerks angezeigt.
Netzwerkkommentar anzeigen	Der Netzwerkkommentar - falls einer definiert ist - wird in der oberen linken Ecke des Netzwerks angezeigt. Wenn außerdem der Netzwerktitel dargestellt wird, erscheint der Kommentar in der Zeile unterhalb.
Operandenkommentar anzeigen	Der Kommentar, der ggfs. einer Variable im Implementierungsteil des Editors hinzugefügt ist, wird angezeigt. Der Operandenkommentar bezieht sich nur auf die lokale Verwendungsstelle der Variable, im Gegensatz zum "Symbolkommentar", der bei der Deklaration der Variable angegeben wrid.
Symbolkommentar anzeigen	Für jedes Symbol (Variable), die bei ihrer Deklaration mit einem Kommentar versehen wurde, wird dieser Kommentar oberhalb des Variablennamens angezeigt. Beachten Sie, dass zusätzlich oder anstelle dessen auch ein lokaler "Operandenkommentar" zugewiesen sein kann.
Symboladresse anzeigen	Für jedes Symbol (Variable) wird die ggfs. zugewiesene Adresse oberhalb des Variablennamens angezeigt.
Feste Größe für Operandenfeld	Wenn diese Option aktiviert ist, definieren die folgenden Parameter die Anzeigegröße der Infromationsfelder für einen Operanden: **Breite des Operandenfeldes [Zeichen]:** Maximale Anzahl an Zeichen des Operandennamens, die dargestellt werden. **Höhe des Operandenfeldes [Zeilen]:** Maximale Anzahl an Zeilen die für die Darstellung des Operandennamens verwendet wird. **Höhe des Operandenkommentarfeldes [Zeilen]:** Maximale Anzahl an Zeilen, die für die Darstellung des Operandenkommentars verwendet werden. **Höhe des Operandensymbolfeldes:** Maximale Anzahl an Zeilen, die für die Darstellung des Operandensymbols verwendet werden.
Bausteinsymbol anzeigen	Wenn ein Funktionsbaustein oder eine Funktion über eine Bibliothek oder über die Objekteigenschaften mit einem Symbol (Bitmap) versehen ist, wird dieses im Bausteinelement im FUP und KOP Editor dargestellt. Auch die Standardoperatoren sind mit Symbolen versehen.  *Abb. 218 Bausteinelemente für Standardoperatoren mit Symbolen in einem FUP-Netzwerk:*

12.23.6 Optionen, Features

In diesem Teil des Optionen-Dialogs kann die Auswahl der Features, die in der aktuellen Programmiersystemumgebung zur Verfügung stehen, angepaßt werden. Im oberen Teil können die einzelnen Funktionalitäten ein- und ausgeschaltet werden, im unteren kann komplett zwischen vordefinierten Feature-Sets gewechselt werden.

Hinweis: Manche Einstellungen schränken die Anzahl der Objekttypen ein, die in einem Projekt neu angelegt werden können. Wenn allerdings ein Projekt geöffnet wird, das bereits Objekte von aktuell nicht mehr einfügbaren Typen enthält, werden diese trotzdem sichtbar und voll funktionsfähig sein.

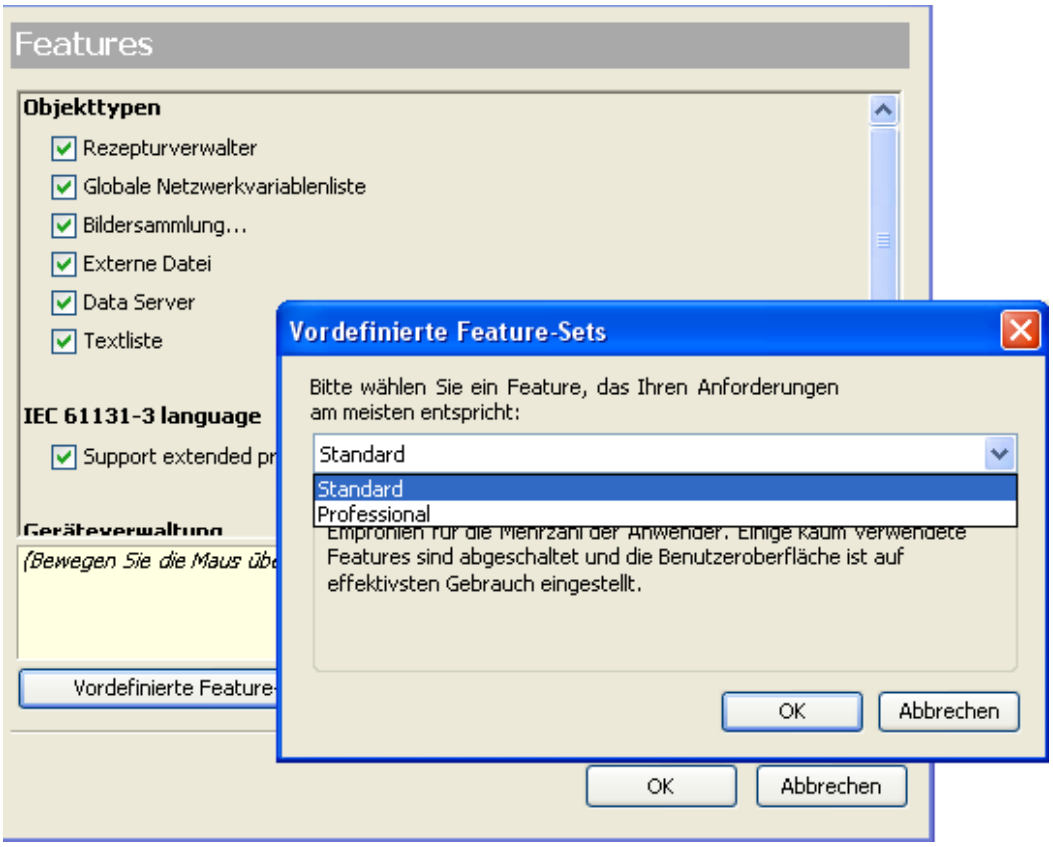


Abb. 219 Dialog Optionen, Features

Vordefinierte Feature-Sets

Zur Zeit gibt es zwei vordefinierte Sets an Funktionalitäten, die über die Schaltfläche **Vordefinierte Feature-Sets** ausgetauscht werden können:

Standard: Einige kaum verwendete Funktionalitäten sind deaktiviert und die Benutzeroberfläche ist auf effektivste Bedienbarkeit ausgelegt. Zu empfehlen für die Mehrheit der Anwender.

Professional: Alle Funktionalitäten sind verfügbar und die Benutzeroberfläche ist komplexer, um alle Möglichkeiten des Systems bereitzustellen.

Einzelne Features

Alternativ kann jedes nur im Professional Set enthaltene Feature einzeln aktiviert (**ein**) oder deaktiviert (**aus**) werden. Zu diesem Zweck setzt man per Mausklick ein Häkchen im jeweiligen Optionsenkästchen. Die betreffenden Features sind in Kategorien eingeteilt, wie in der folgenden Tabelle gezeigt. In den letzten beiden Spalten ist die Standardeinstellung für das Standard bzw. Professional Set angegeben.

Kategorie	Beschreibung	Standard	Professional
Objekttypen:			
Data Server Bildersammlung Externe Datei Rezepturverwalter Textliste Globale Netzwerkvariablenliste	Nur wenn das Häkchen für ein Objekt gesetzt ist, wird der Objekttyp im "Objekt hinzufügen " Kontextmenü erscheinen. Textliste und Bildersammlung werden außerdem bei Bedarf automatisch durch die Visualisierung angelegt.	off	on
IEC 61131-3 Sprache:			
Erweiterte Programmierfunktionalitäten unterstützen:	Nur wenn dieses Feature aktiviert ist, werden die folgenden Objekttypen im "Objekt hinzufügen " Kontextmenü erscheinen: - Schnittstelle - Methode - Eigenschaft Außerdem wird der Assistent beim Anlegen einer neuen Funktionsbaustein-POU das Erweitern (EXTENDS) und Implementieren von Schnittstellen (IMPLEMENTS) anbieten. Der tabellarische Deklarationseditor wird ebenfalls das Erweitern von Funktionsbausteinen, das Implementieren von Schnittstellen; zusätzlich können eine POU oder einzelne Deklarationen mit Compiler-Pragmas versehen werden. Die Eingabehilfe wird Schlüsselwörter anbieten, die für objektorientierte Programmierung verwendet werden (EXTENDS, IMPLEMENTS, etc.)	off	on
Geräteverwaltung			
Vereinfachtes Geräte-Handling aktivieren	Wenn dieses Feature aktiviert ist, sieht der Anwender nur die neueste Version der Gerätebeschreibung für ein Gerät; die Version selbst wird nur an einigen Stellen zur Information angezeigt.	on	off

Kategorie	Beschreibung	Standard	Professional
Repository-Dialog aktivieren	Nur wenn dieses Feature aktiviert ist, ist der Menübefehl Tools / Geräte-Repository... verfügbar. Ansonsten erscheint statt dessen der Befehl 'Tools / Gerät installieren...' , der zum Dialog Gerät installieren führt. Alternativ können neue Gerätebeschreibungen über den Package-Mechanismus installiert werden.	off	on
Bibliotheksverwaltung			
Vereinfachtes Bibliotheks-Handling aktivieren	Wenn dieses Feature aktiviert ist, gibt es folgende Vereinfachungen: - Ein Projekt kann nicht als Bibliothek gespeichert werden. - Ein Projekt kann nicht ins Bibliotheks-Repository installiert werden. - Bibliotheksprojekte können nicht zur Bearbeitung geöffnet werden. - Wenn eine Bibliothek im Bibliotheksverwalter eingefügt wird, können keine Platzhalter-Referenzen hinzugefügt werden. - In den Eigenschaften einer Bibliotheksreferenz kann nur der Namensraum bearbeitet und geändert werden. - Im Bibliotheksverwalter selbst wird die Spalte "Version" nicht angezeigt. - Wenn eine Bibliothek im Bibliotheksverwalter eingefügt wird, wird nur die neueste Version pro Bibliothek angeboten, die Versionsangabe selbst erscheint nicht.	on	off
Systembibliotheken verstecken	Wenn dieses Feature aktiviert ist, werden Systembibliotheken (ansonsten in grauer Schrift dargestellt) nicht im Bibliotheksverwalter dargestellt. Ausnahme: Wenn eine solche Bibliothek nicht geladen werden konnte wird sie dargestellt, damit der Anwender die Referenz manuell entfernen kann.	on	off
Repository-Dialog aktivieren	Nur wenn dieses Feature aktiviert ist, ist der Menübefehl Tools / Bibliotheks-Repository... verfügbar. Ansonsten erscheint statt dessen der Befehl 'Tools / Bibliothek installieren...' , der zum Dialog Bibliothek auswählen führt. Alternativ können neue Bibliotheken über den Package-Mechanismus installiert werden.	off	on

12.23.7 Optionen, Geräteeditor

Dieser Unterdialog des Optionen-Dialogs ermöglicht allgemeine Einstellungen für das Arbeiten im Geräteeditor.

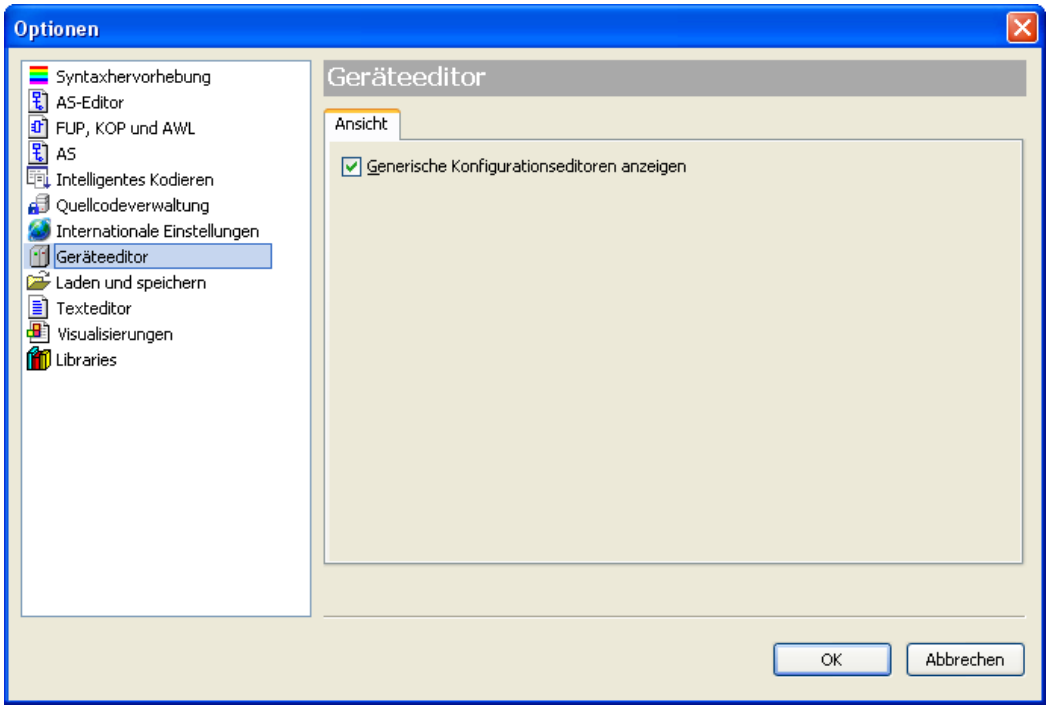


Abb. 220 Dialog Geräteeditor-Optionen

Der Unterdialog 'Ansicht' enthält die Einstellung **Generische Konfigurationseditoren anzeigen**. Wenn diese Option aktiviert ist, ist für parametrierbare Geräte der 'Konfiguration's-Dialog im Geräteeditor verfügbar.

12.23.8 Optionen, AS-Editor

Dieser Unterdialog des Optionen-Dialogs enthält verschiedene Registerblätter, wo verschiedene Kategorien von Einstellungen für den AS (SFC)-Editor vorgenommen werden können.

Anordnung

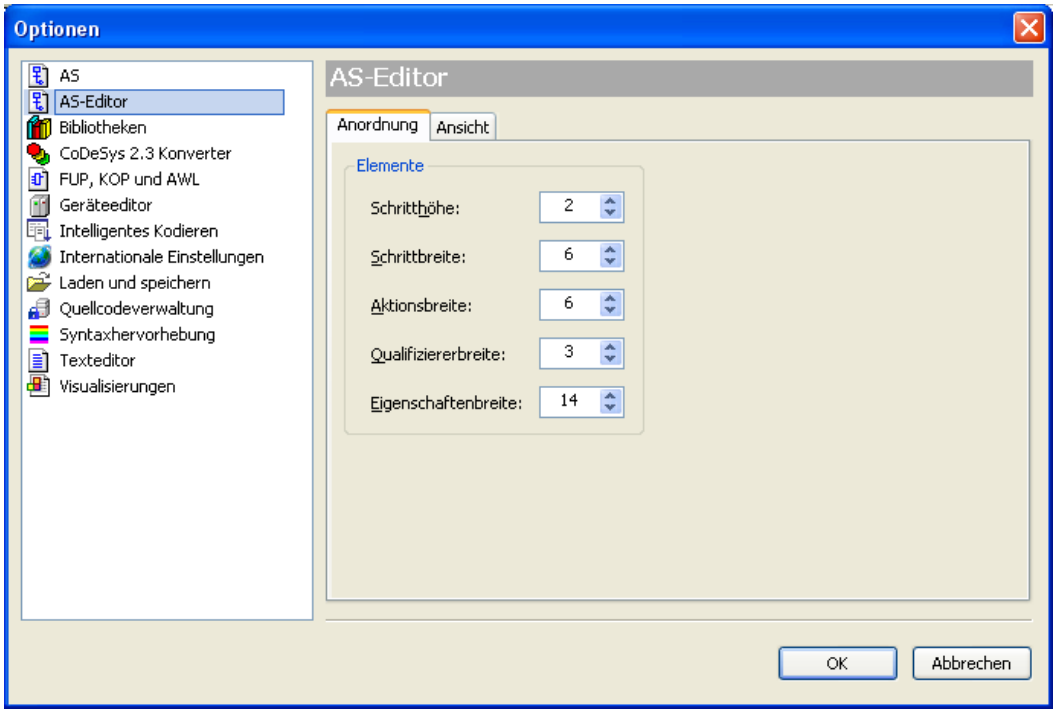


Abb. 221 AS-Editor Optionen-Dialog, Kategorie Anordnung

Die Werte werden in "Rastereinheiten" angegeben. 1 Rastereinheit = Fontgröße, die aktuell in den Texteditor-Optionen gesetzt ist (Textbereich / Schriftart).

Schritthöhe: Höhe eines Schritt-Elements. Mögliche Werte: 1-100.

Schrittbreite: Breite eines Schritt-Elements. Mögliche Werte: 2-100.

Aktionsbreite: Breite eines Aktionen-Elements. Mögliche Werte: 2-100.

Qualifiziererbreite: Breite des Qualifizierers. Mögliche Werte: 2-100.

Eigenschaftenbreite: Breite des Anzeigebrereichs für die Schritt-Eigenschaften. Mögliche Werte: 2-100.

Die Einstellungen werden immer sofort in allen gerade geöffneten AS-Editorfenstern wirksam.

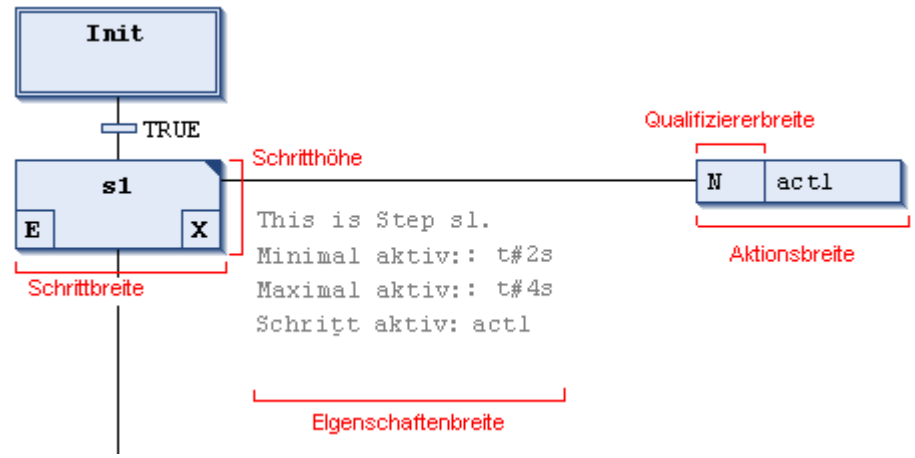


Abb. 222 Einstellungsparameter für die Anordnung in AS

Ansicht

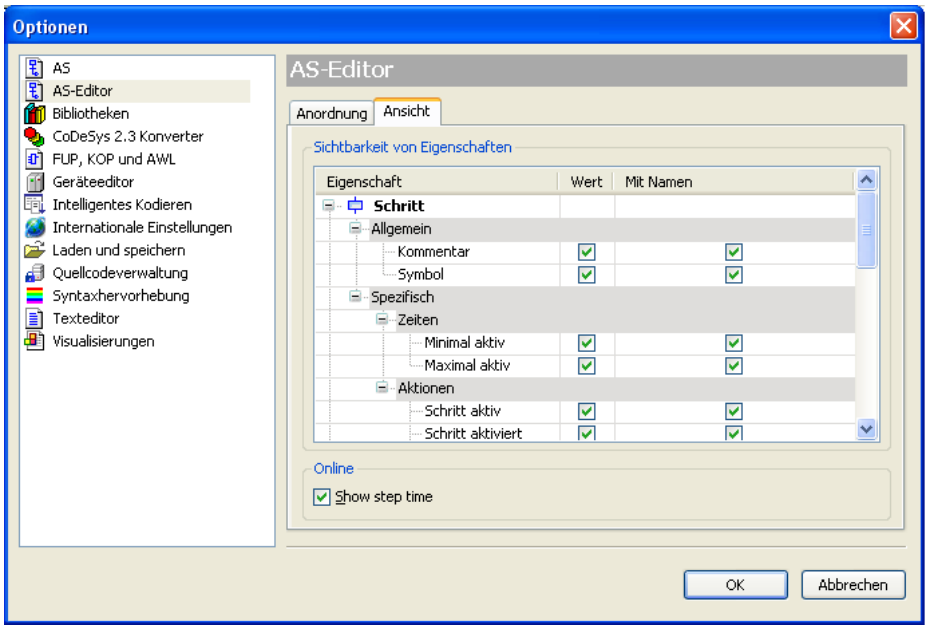
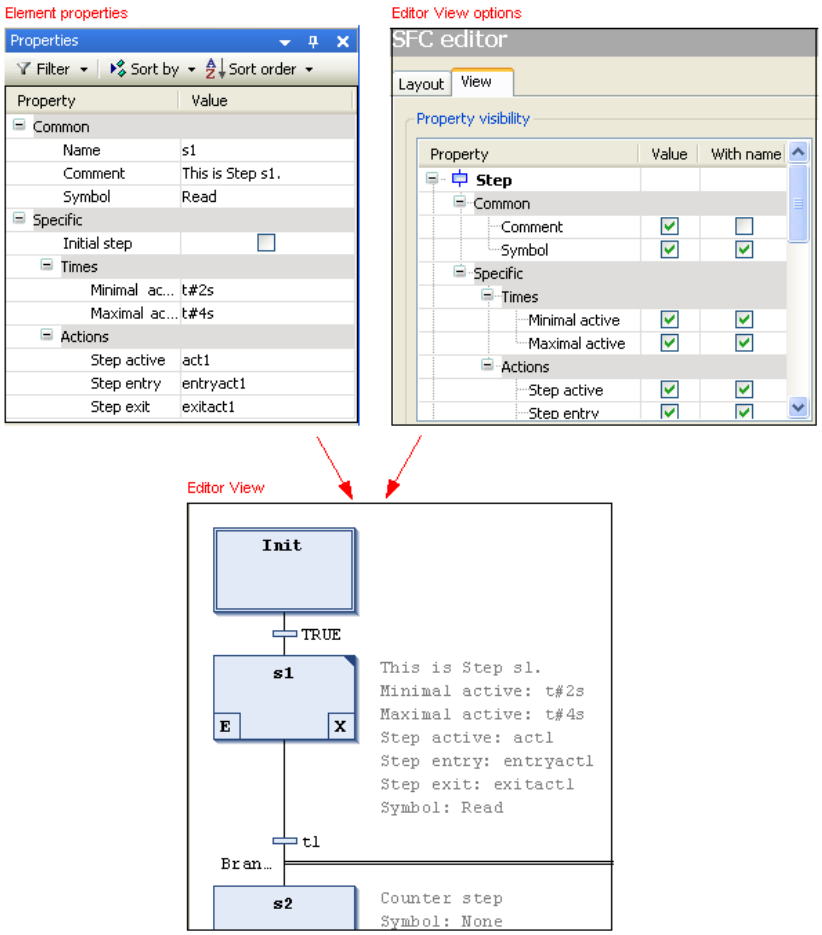


Abb. 223 AS-Editor Optionen-Dialog, Kategorie Ansicht

Eigenschaft:

Hier kann definiert werden, welche der Element-Eigenschaften neben dem Element im AS-Diagramm dargestellt werden sollen. Eigenschaften der Kategorie 'Allgemein' und 'Spezifisch' werden in der Tabelle aufgelistet. Sie können für jede einzeln ein Häkchen in der Spalte **Wert** setzen, wenn der Eigenschaften-Wert angezeigt werden soll und zusätzlich ein Häkchen in Spalte **Mit Namen**, wenn vor dem Wert auch noch der Eigenschaften-Name angezeigt werden soll.



Online:

Wenn die Option **Schrittzeit anzeigen** aktiviert ist, wird im Online-Modus die aktuelle Schrittzeit rechts von allen Schritten angezeigt, für die Zeit-Eigenschaften definiert sind.

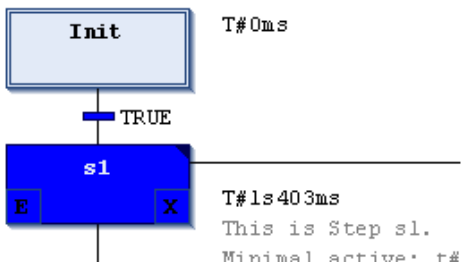


Abb. 224 Beispiel der Anzeige einer Schrittzeit im Online-Modus

12.23.9 Optionen, AS

In diesem Unterdialog des Optionen-Dialogs können **Standardeinstellungen für AS-Objekte** definiert werden. Jedes **neue** AS-Objekt wird diese Einstellungen dann automatisch in seinen Eigenschaften enthalten.

Hinweis: Auf bereits bestehende AS-Objekte müssen hier vorgenommene Änderungen der Einstellungen explizit über die Schaltfläche 'Voreinstellungen' im Objekt-Eigenschaften-Dialog übertragen werden.

Beachten Sie, dass die reinen Editor-Optionen, d.h. Einstellungen für Layout und Ansicht, in einem separaten Optionen-Dialog gehandhabt werden.

Übersetzen

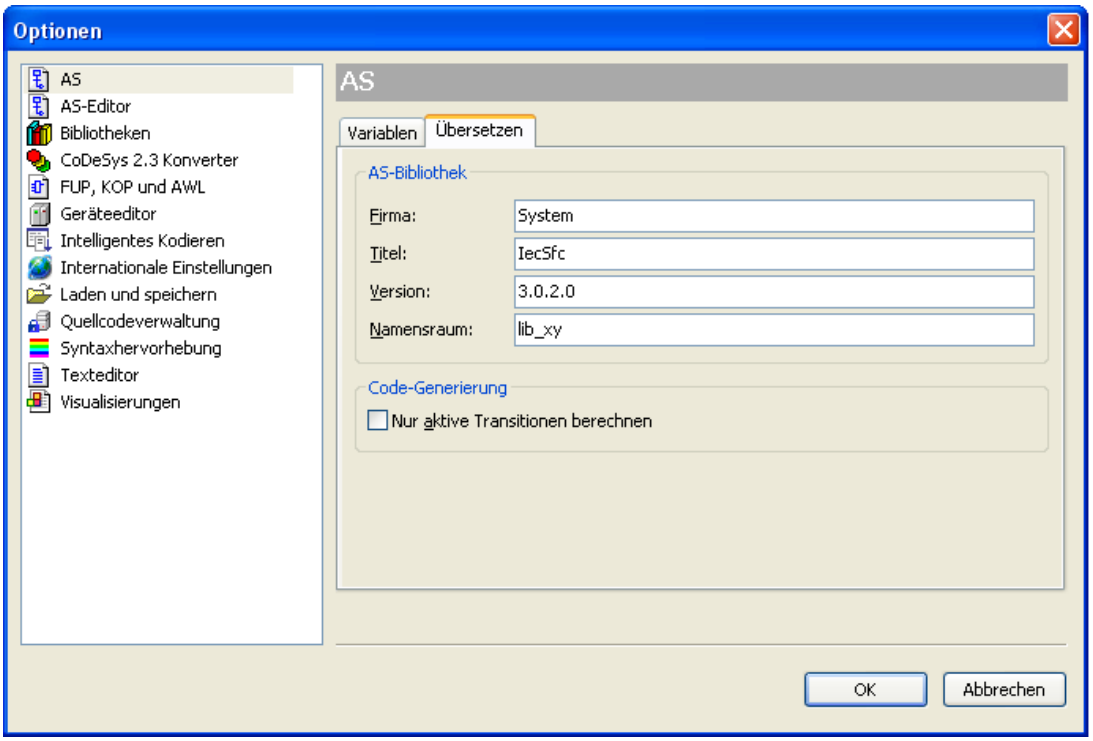


Abb. 225 AS Optionen, Kategorie 'Übersetzen'

AS-Bibliothek:

Hier wird die AS-Bibliothek definiert, die per Default für neue AS-Objekte verwendet werden soll. **Firma**, **Titel** und **Version** werden wie in den Projekteinstellungen der Bibliothek mitgeliefert, angegeben. Ein **Namensraum** kann zusätzlich angegeben werden um eine Bibliothek eindeutig anzusprechen. Dies könnte nötig sein, wenn mehrere Versionen der Bibliothek auf dem System verfügbar sind. Achten Sie bei der Eingabe eines Namensraumes jedoch immer darauf, dass der hier angegebene mit dem übereinstimmt, der im Bibliotheksverwalter für die Bibliothek definiert ist!

Standardmäßig wird dieser Dialog mit den Einstellungen für die Bibliothek **SfcIec.library** vorbelegt, die mit dem Standard-Profil mitgeliefert wird.

Code-Generierung:

Nur aktive Transitionen berechnen: Wenn diese Option aktiviert ist, wird nur für die gerade aktiven Transitionen Code erzeugt.

Variablen

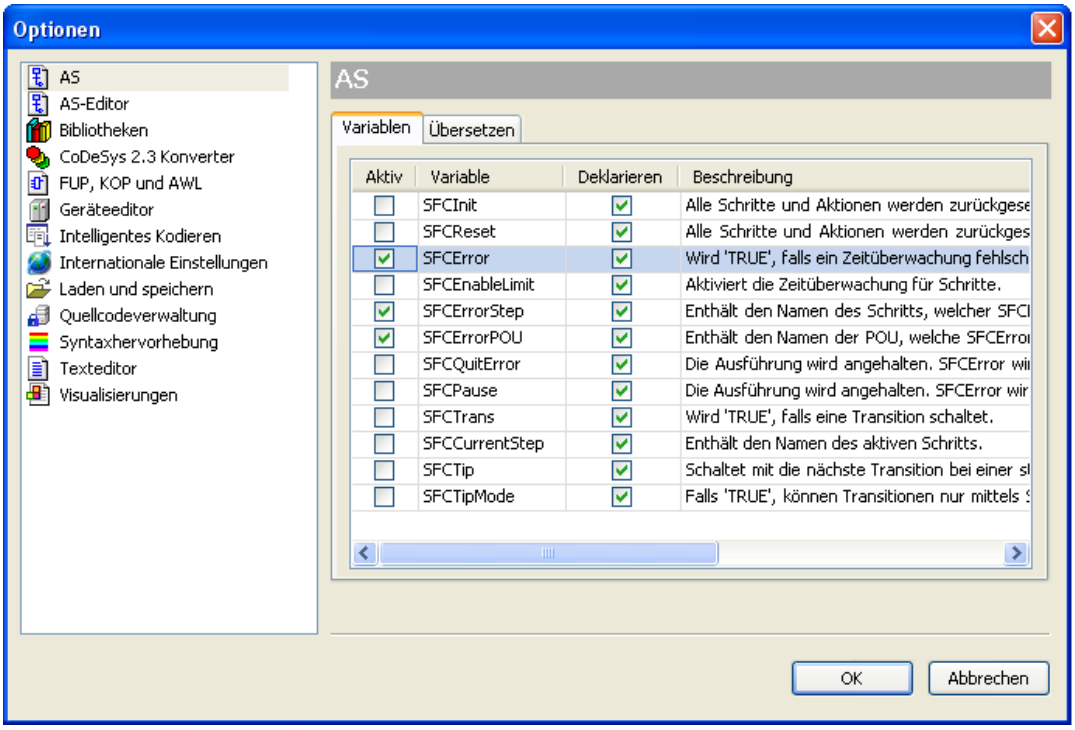


Abb. 226 AS Optionen, Kategorie 'Variablen'

Alle möglichen 'Flags', d.h. implizit erzeugten Variablen für das Überwachen und Kontrollieren der Abarbeitung in einem AS-Diagramm, werden hier aufgelistet. Eine kurze Beschreibung ist im Dialog beigefügt.

Mit einem Mausklick auf das Kontrollkästchen kann für jede Flag-Variable die automatische Deklaration (**Deklarieren**) und Verwendung (**Aktiv)** aktiviert werden. Diese Einstellungen werden dann als Standardeinstellungen für neue AS-Objekte übernommen.

Wenn 'Deklarieren' aktiviert ist, aber 'Active' nicht, dann wird zwar die Deklaration der Variable erfolgen, aber das Flag hat während der Abarbeitung keine Funktion.

Beachten Sie, dass eine automatisch deklarierte Flag-Variable **nur im Online-Modus** im Deklarationsteil des AS-Editors sichtbar ist!

In Schritt 's1' in AS Objekt 'POU' wurde durch das Flag SFCError eine Zeitüberschreitung registriert.

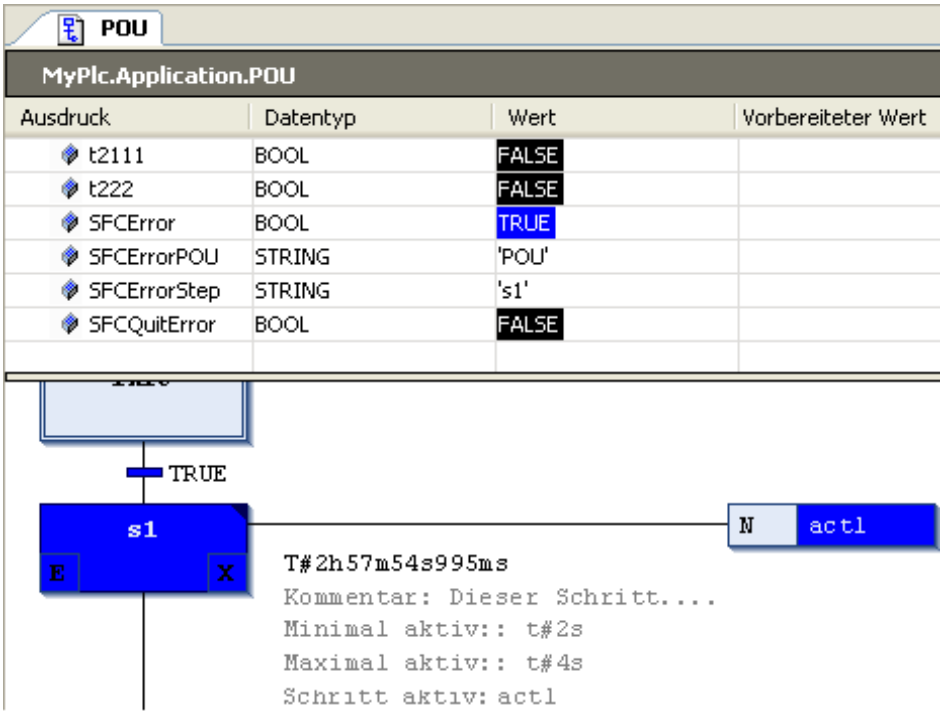


Abb. 227 Beispiel einiger AS Error-Flags im Online-Modus des Editors

12.23.10 Optionen, Quellcodeverwaltung

In diesem Unterdialog des Optionen-Dialogs können allgemeine Einstellung für die Quellcodeverwaltung des Projekts vorgenommen werden.

Die Quellcodeverwaltung für ein Projekt muss explizit aktiviert werden (im Dialog 'Quellcodeverwaltung ändern') und um sie in Anspruch zu nehmen, muss man sich auf dem ENI Server anmelden, der die Verbindung zur externen Projektdatenbank herstellt.

Im vorliegenden Dialog wird definiert, welche Aktionen bezüglich der Quellcodeverwaltung automatisch ablaufen sollen. Stellen Sie für jede der Aktionen ein, ob sie ohne Nachfrage ausgeführt werden soll (**Ja**), ob vor der Ausführung nachgefragt werden soll (**Nachfragen**), oder ob sie überhaupt nicht ausgeführt werden soll (**Nein**).

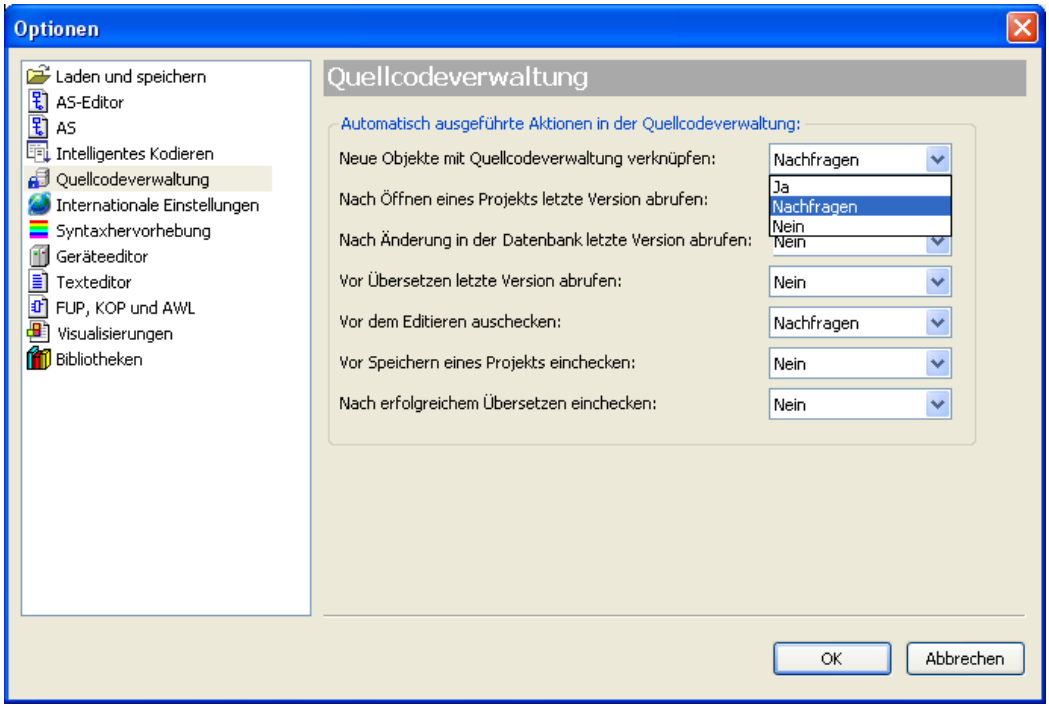


Abb. 228 Quellcodeverwaltung, Optionen-Dialog

Die Aktionen:

Neue Objekte mit Quellcodeverwaltung verknüpfen	Wenn dem Projekt ein neues Objekt hinzugefügt wird, wird dieses mit dem Standard-Speicherort in der Projektdatenbank verknüpft.
Nach Öffnen eines Projekts letzte Version abrufen	Wenn das Projekt geöffnet wird, wird die letzte Version aller unter Quellcodeverwaltung befindlichen Objekte aus der Datenbank ins Projekt kopiert. Dabei wird die lokale Version der Objekte überschrieben.
Nach Änderung in der Datenbank letzte Version abrufen	Sobald eine neuere Version eines Objekts in der Datenbank eingecheckt wird (z.B. durch einen anderen Benutzer), wird der lokale Benutzer informiert.

Vor Übersetzen letzte Version abrufen	Wenn ein Übersetzen des Projekts ansteht, wird zunächst die letzte Version aller unter Quellcodeverwaltung befindlichen Objekte aus der Datenbank ins Projekt kopiert. Dabei wird die lokale Version der Objekte überschrieben.
Vor dem Editieren auschecken	Sobald mit dem Editieren eines unter Quellcodeverwaltung befindlichen Objekts begonnen wird, wird das Objekt ausgecheckt. Wenn das Objekt gerade durch einen anderen Benutzer ausgecheckt ist (angezeigt durch ein rotes Kreuzchen vor dem Objektnamen im POUs- oder Gerätebaum), wird eine entsprechende Meldung ausgegeben.
Vor Speichern eines Projekts einchecken	Vor dem Speichern des Projekts werden alle unter Quellcodeverwaltung befindlichen Objekte eingecheckt.
Nach erfolgreichem Übersetzen einchecken	Nach einem fehlerfreien Übersetzungslauf des Projekts werden alle unter Quellcodeverwaltung befindlichen Objekte eingecheckt.

12.23.11 Optionen, Visualisierungen

Dieser Unterdialog des Optionen-Dialogs besteht aus den Registerblättern **Allgemein** und **Raster**, die Einstellungen für das Arbeiten mit Visualisierungen im Programmiersystem erlauben.

Allgemein

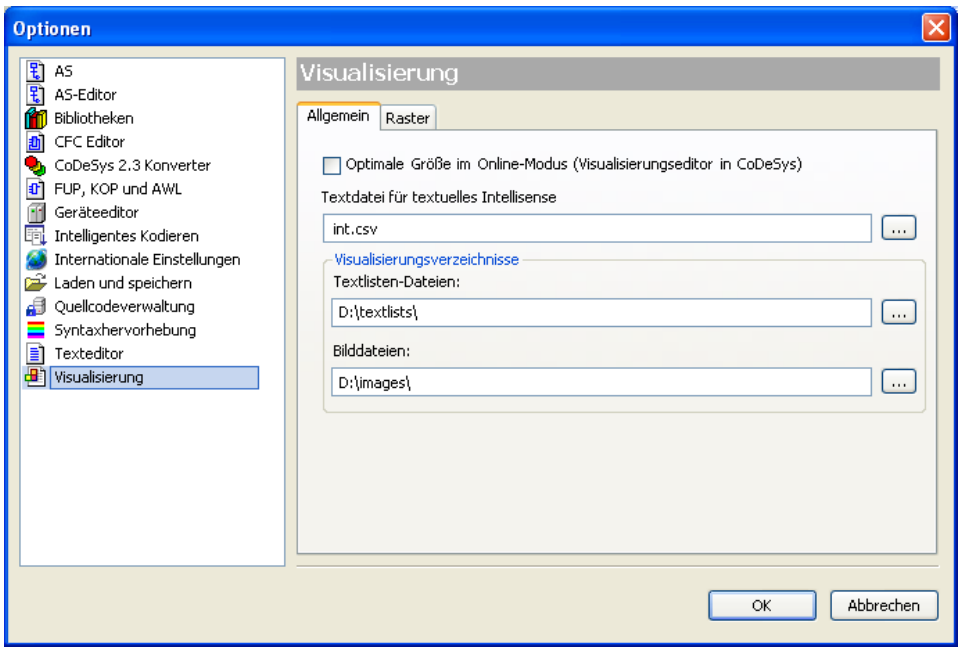


Abb. 229 Dialog für Visualisierungsoptionen, Allgemein

Anpassen im Online Modus: Diese Option muss aktiviert werden, wenn die Visualisierung im Online-Betrieb an die aktuelle Editorfenster-Größe angepaßt werden soll, so dass immer alle Elemente sichtbar sind.

Hinweis: Die Skalierung für eine Targetvisualisierung kann innerhalb des Visualzation Manager Editors definiert werden.

Textdatei für textuelles "Komponenten auflisten": Im darunter stehenden Eingabefeld kann der Name (inklusive Pfad) einer Datei (vom Format .csv) eingetragen werden. Diese dient dann als Quelle für die Vorschläge, welche die **"Komponenten auflisten"** Funktion enthält beim Editieren des Feldes 'Texts\Text' innerhalb der Eigenschaften eines Visualisierungselementes. Diese Textdatei kann beispielsweise aus dem Export einer globalen Textliste hervorgehen. Ihr Aufbau ist tabellenförmig und muss der Struktur von Textlisten folgen, die auf der Hilfeseite zu Textlisten beschrieben ist.

Visualisierungsverzeichnisse:

Hier können Standard-Verzeichnispfade vordefiniert werden, die in den Projekteinstellungen, Kategorie Visualisierung jedoch projektspezifisch noch verändert werden können. Letztendlich bestimmend für das Projekt sind die Projekteinstellungen!

Textlisten-Dateien: Öffnen Sie mit Schaltfläche [...] den Standarddialog zur Definition eines existierenden oder eines neuen Verzeichnisses für die in den Visualisierungen verwendeten Sprachdateien (Textlisten).

Bilddateien: Öffnen Sie mit Schaltfläche [...] den Standarddialog zur Definition eines existierenden oder eines neuen Verzeichnisses für die Bilddateien (Bildersammlungen), die im Projekt für die Verwendung in Visualisierungen bereitstehen sollen. Hier können mehrere Verzeichnisse in einer Strichpunkt-separierten Liste angegeben werden.

Raster

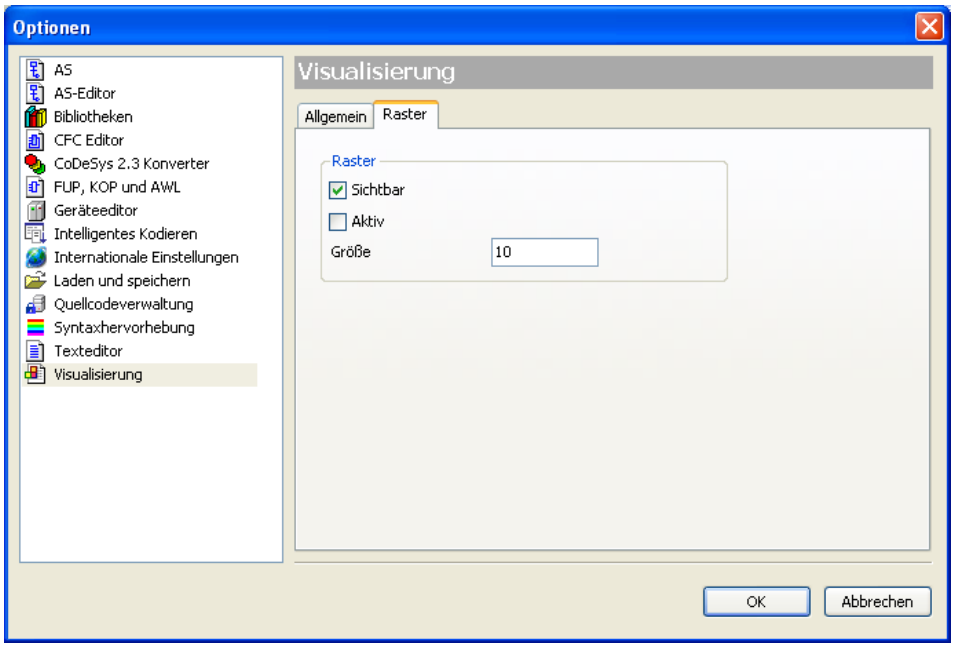


Abb. 230 Dialog für Visualisierungsoptionen, Raster

Sichtbar: Wenn diese Option aktiviert ist, wird ein Raster gemäß der unten definierten Größe als optische Orientierungshilfe im Visualisierungseditor eingeblendet.

Aktiv: Wenn diese Option aktiviert ist, können Visualisierungselemente nur auf den Punkten des unten in "Größe" definierten Rasters positioniert werden, egal ob das Raster "sichtbar" oder nur "aktiv" gesetzt ist. Beim Einfügen oder Verschieben eines Elements bedeutet dies, dass das Zentrum des Elements nur auf einem Rasterpunkt platziert werden kann. Beim Verändern eines Elements bedeutet es, dass das kleine Rechteck an der Elementkante, das bewegt wird, um die Größe oder Form des Elements zu ändern, nur auf einen Rasterpunkt verschoben werden kann.

Größe: Vertikaler und horizontaler Abstand zwischen den Rasterpunkten in Pixel.

12.23.12 Anpassen-Dialoge

Der Dialog 'Anpassen', der mit dem Plug-In 'Frame' installiert wird, enthält Unterdialoge (Registerblätter) für die Konfiguration der Benutzeroberfläche in Bezug auf die Menü-Struktur, die Symbolleisten und die Tastaturbedienung.

Der Dialog wird mit dem Befehl '**Anpassen**' (Kategorie 'Einstellungen') geöffnet, der standardmäßig im 'Tools'-Menü enthalten ist.

Die Standardeinstellungen werden vom Hersteller definiert und können jederzeit über die Schaltfläche **Zurücksetzen** wieder hergestellt werden. Ein Dialog öffnet sich in diesem Fall, in dem man auswählen kann, welche der Einstellungs-Kategorien Menü, Tastatur, Symbolleisten auf die Standardwerte zurückgesetzt werden soll.

Die aktuellen Einstellungen werden mit OK bestätigt und auf dem lokalen Rechner gespeichert. Sie können auch explizit als Konfigurationsdateien gespeichert werden, die später wieder geladen werden können, siehe unten.

Die folgenden Unterdialoge sind per Default verfügbar:

- Menü
- Tastatur
- Symbolleisten

Speichern und Laden der Einstellungen:

Mit Schaltfläche **Speichern** erhalten Sie den Standarddialog zum Speichern einer Datei für die Einstellungen des gerade geöffneten Unterdialogs. Geben Sie hier einen Dateinamen ohne Erweiterung an und bestätigen mit 'Speichern'. Dies kann für jeden der drei Unterdialoge durchgeführt und somit die folgenden Konfigurationsdateien gespeichert werden:

"<dateiname>.opt.menu" für die Menüeinstellungen

"<filename>.opt.keyb" für die Tastatureinstellungen

"<filename>.opt.tbar" für die Symbolleisten-Einstellungen

Um die Einstellungen aus einer gespeicherten Konfigurationsdatei wieder zu laden, verwenden Sie im jeweiligen Unterdialog die Schaltfläche **Laden**. Der Standarddialog zum Öffnen einer Datei erscheint, wo der Filter bereits auf die passenden Konfigurationsdateien gesetzt ist. Nach Auswahl und Öffnen der gewünschten Datei werden die Einstellungen in den Dialog übernommen.

Symbolleisten anpassen

Dieser Unterdialog des Anpassen-Dialogs erlaubt

- - das Aktivieren und Deaktivieren von Symbolleisten (Toolbars), d.h. das Ein- oder Ausblenden dieser Leisten am oberen Rand der Benutzeroberfläche
- - das Hinzufügen neuer Symbolleisten, die im Dialog 'Symbolleiste bearbeiten..' definiert wurden.
- - das Löschen von Symbolleisten
- - das Speichern der aktuellen Symbolleisten-Definitionen in eine externe Datei bzw. das Laden einer solchen extern abgelegten Konfigurationsdatei
- - das Umbenennen von Symbolleisten

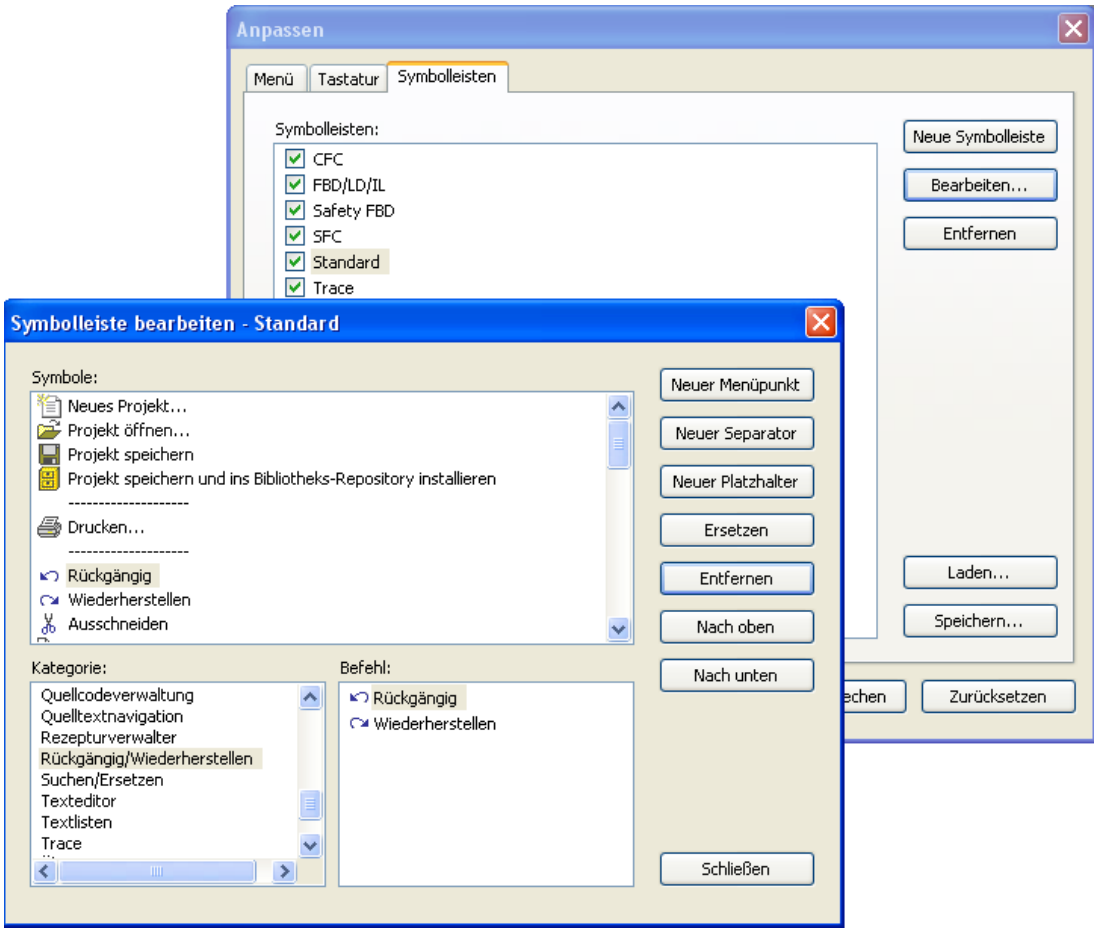


Abb. 231 Dialoge zum Anpassen der Symbolleisten

Symbolleiste aktivieren:

Die 'Toolbars:'-Liste zeigt die Namen der aktuell definierten Symbolleisten.

Um eine Symbolleiste zu aktivieren bzw. deaktivieren, d.h. sie in der Oberfläche ein- bzw. auszublenden, klicken Sie mit der Maus auf das jeweilige Kontrollkästchen um einen Haken zu setzen (aktiviert) bzw. zu entfernen (deaktiviert).

Symbolleiste definieren:

Verwenden Sie die Schaltfläche **Neue Symbolleiste**. Ein neuer Eintrag wird an oberster Position in die Liste der Symbolleisten eingefügt, mit einem geöffneten Eingabefeld für den Namen der neuen Leiste.

Um nun die neue Symbolleiste zu gestalten, verwenden Sie Schaltfläche **Bearbeiten...**. Der **Dialog 'Symbolleiste bearbeiten - <Symbolleistenname>'** öffnet, wo definiert wird, welche Befehle in der Symbolleiste enthalten sein sollen:

Wählen Sie einen Befehl aus der Liste 'Befehl' aus, die immer alle Befehle der gerade im Fenster 'Kategorie' gewählten Befehlskategorie anzeigt. Dann drücken Sie Schaltfläche **Neues Symbol**, um den Befehl in der Liste der **Symbole** im oberen Teilfenster des Dialogs hinzuzufügen, d.h. es in die Symbolleiste aufzunehmen. Auf diese Weise fügen Sie alle gewünschten Kommandos hinzu. Die Anordnung in der Symbole-Liste entspricht der, die später in der Symbolleiste gilt. Um sie zu verändern, können Sie die Schaltflächen **Nach unten**, **Nach oben** und **Entfernen** verwenden. Zur optischen Strukturierung können Sie Trennlinien einfügen (**Neuer Separator**).

Platzhalter verwenden:

Ein Platzhalter in der Definition einer Symbolleiste repräsentiert eine bestimmte Befehlskategorie. Dies ist momentan ohne Auswirkung auf das Aussehen des Menüs in der Benutzeroberfläche, kann bei späteren Erweiterungen von Befehlskategorien nützlich sein. Wenn später ein neues Plug-In neue Befehle für die durch den Platzhalter vertretene Kategorie A bringt, werden diese automatisch an der Position des Platzhalters in der Symbolleiste eingefügt. Ein Platzhalter kann über die Schaltfläche **Neuer Platzhalter** oberhalb des gerade im Menüstruktur-Fenster gewählten Eintrags eingefügt werden.

Symbolleiste löschen:

Um eine Symbolleiste und ihre Definition komplett aus der Konfiguration zu löschen, wählen Sie sie in der Liste aus und verwenden Schaltfläche **Entfernen**.

Symbolleiste umbenennen:

Wenn Sie eine Symbolleiste innerhalb der Konfiguration anders nennen wollen, können Sie den Namen direkt in der Liste editieren. Um in den Editier-Modus zu gelangen, wählen Sie den Eintrag in der Liste aus und drücken die <Leertaste> oder führen einen weiteren Mauskllick auf den Eintrag aus.

Menü anpassen

Dieser Unterdialog des Anpassen-Dialogs ermöglicht das Definieren der Struktur und der Inhalte der Menüs für die Benutzeroberfläche.

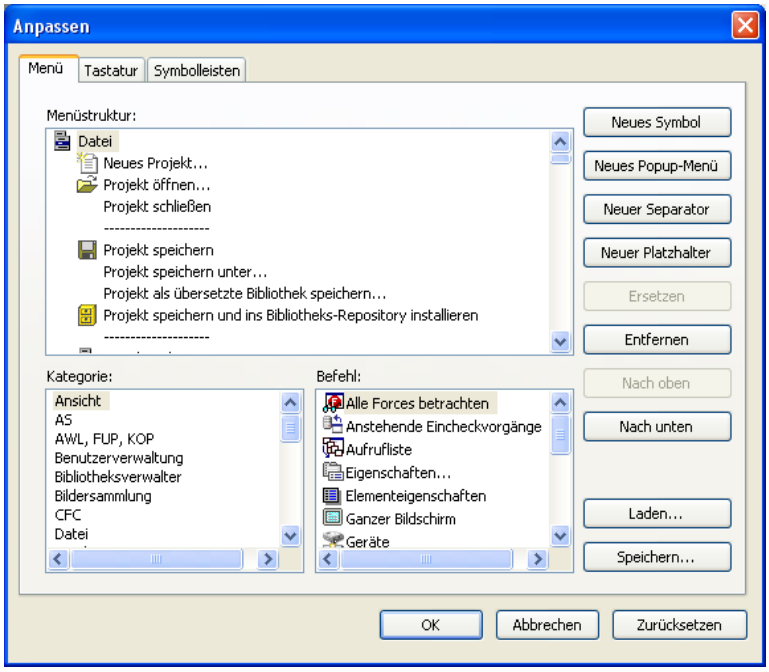


Abb. 232 Anpassen-Dialog, Menü

Menüstruktur Fenster: Die aktuell definierten Menüs, Untermenüs und die enthaltenen Befehle werden dargestellt. Ein Menü- oder Untermenü-Titel wird durch das Symbol gekennzeichnet. Darunter eingerückt stehen die zugehörigen Befehle und Untermenüs in der gleichen Reihenfolge, wie sie dann in der Benutzeroberfläche erscheinen werden.. Trennlinien zur optischen Strukturierung können eingefügt sein.

Kategorie Fenster: Hier finden Sie die Kategorien von Befehlen, die das aktuelle Profil bereitstellt. Jede Kategorie enthält ein oder mehrere Befehle, die im **Befehl**-Fenster immer für die gerade ausgewählte Kategorie angezeigt werden.

Neuen Befehl oder Separator in ein bestehendes Menü einfügen:

Im **Menüstruktur**-Fenster wählen Sie den Titel des Menüs aus, dem sie einen Befehl hinzufügen wollen. Im **Kategorie-Fenster wählen sie die gewünschte Kategorie und dann den gewünschten Befehl im Befehl-Fenster. Wenn Sie dann Schaltfläche Neues Symbol wählen, wird der Befehl im Menüstruktur-Fenster unterhalb des Menütitels eingefügt. Um es innerhalb des Menüs zu verschieben, können Sie die Schaltflächen Nach unten oder Nach oben verwenden.**

Alternativ kann auch ein bereits vorhandener Befehl **im Menüstruktur-Fenster ausgewählt werden. Der neue Befehl wird dann direkt oberhalb von diesem eingefügt.**

Mit Schaltfläche Neuer Separator wird eine Trennlinie oberhalb des gerade im Menüstruktur-Fenster ausgewählten Befehls eingefügt.

Neues Menü oder Untermenü definieiren:

Wählen Sie im **Menüstruktur**-Fenster den Eintrag aus, oberhalb dessen ein neues Menü oder Untermenü eingefügt werden soll. Die Anordnung im Menüstruktur-Fenster von oben nach unten entspricht der später in der Benutzeroberfläche verwendeten Anordnung von links nach rechts. Über Schaltfläche **Neues Popup-Menü** wird der Dialog **Objekt bearbeiten** geöffnet, in dem ein Menüname in verschiedenen Landessprachen definiert werden kann:

Abb. 233 Dialog 'Objekt bearbeiten' für Menütitel

Der **Standardtext** wird als Menütitel verwendet, wenn für die aktuelle Spracheinstellung kein lokalisierter Text definiert ist. Ersetzen Sie "<Popup>" durch den für diesen Fall gewünschten Menünamen. Für eine Lokalisierung können Sie bei **Sprache** aus der Auswahlliste verschiedene Landessprachen auswählen und jeweils dafür im Feld **Lokalisierter Text** den Menünamen in der jeweiligen Landessprache eingeben. Es können Definitionen für mehrere Sprachen hintereinander eingegeben werden. Alle Einstellungen werden dann beim Schließen gespeichert.

Das neue Menü wird allerdings nur dann in der Menüleiste angezeigt, wenn ihm mindestens ein Befehl zugewiesen wurde. Der Menütitel erscheint in der gerade im Programmiersystem eingestellten Landessprache wenn dafür ein lokalisierter Text eingetragen wurde.

Platzhalter einfügen:

Platzhalter verwenden:

Ein Platzhalter in der Definition einer Symbolleiste repräsentiert eine bestimmte Befehlskategorie. Dies kann bei späteren Erweiterungen von Befehlskategorien nützlich sein. Wenn ein neues Plug-In neue Befehle für die durch den Platzhalter vertretene Kategorie A bringt, werden diese automatisch an der Position des Platzhalters in der Symbolleiste eingefügt. Ein Platzhalter kann über die Schaltfläche **Neuer Platzhalter**.

Beispiel: Im Menüstruktur-Fenster wählen Sie Schaltfläche 'Bearbeiten', im Kategorie-Fenster wählen sie 'Zwischenablage' und drücken nun Schaltfläche 'Neuer Platzhalter'. Der Platzhalter "<Zwischenablage>" wird unterhalb vom Eintrag 'Bearbeiten' in der Menüstruktur eingefügt:

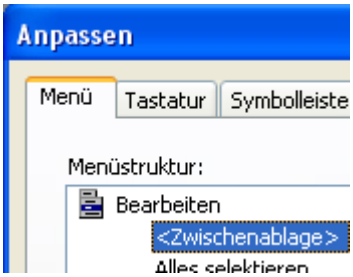


Abb. 234 Beispiel, Platzhalter in Menü eingefügt

Dies hat zunächst keinerlei Auswirkung auf das Aussehen des 'Bearbeiten'-Menüs in der Benutzeroberfläche. Erst wenn ein neues Plug-In verwendet wird, das einen neuen Befehl (z.B. "Nichts auswählen") der Befehlskategorie 'Zwischenablage' liefert, wird dieser Befehl im 'Bearbeiten'-Menü an oberster Stelle angezeigt werden.

Tastatur anpassen

Dieser Unterdialog des Anpassen-Dialogs ermöglicht, Tastaturkürzel zu definieren, d.h. Tasten oder Tastenkombinationen mit den verfügbaren Befehlen aus den diversen Befehlskategorien zu belegen.

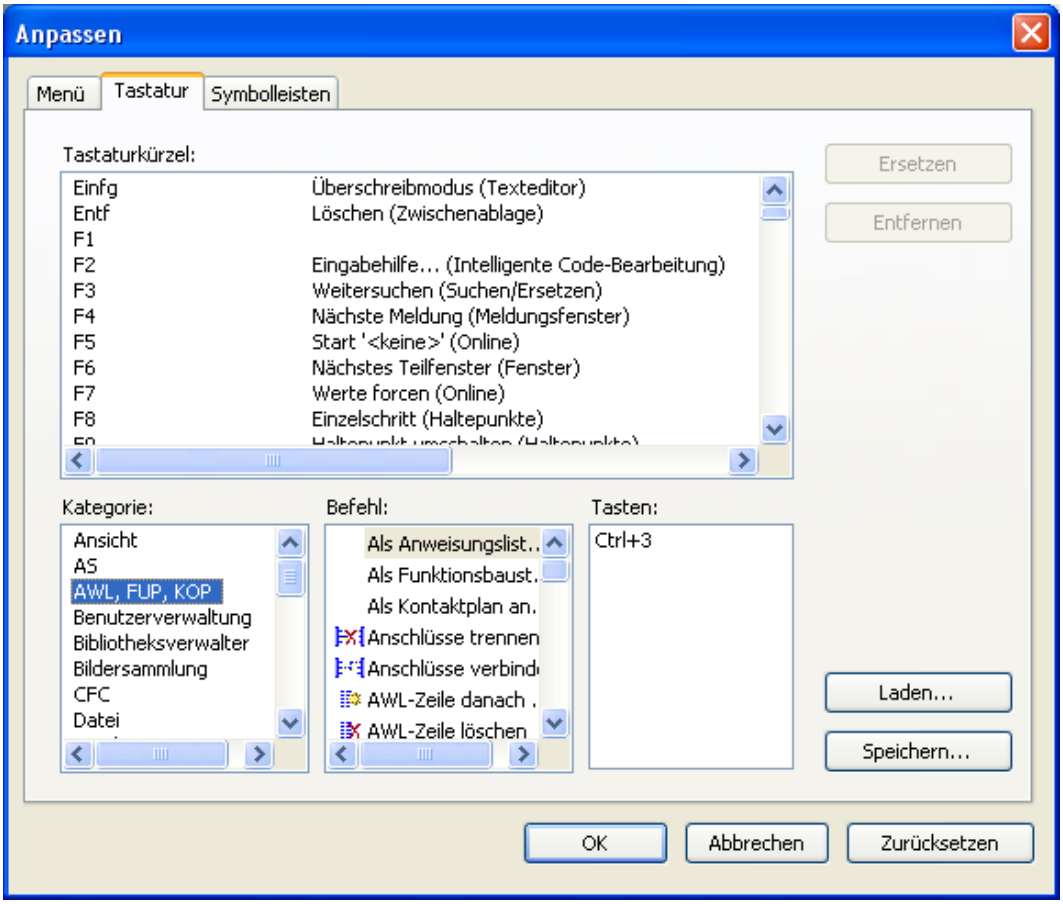


Abb. 235 Anpassen-Dialog, Tastatur

Tastaturkürzel: Dieses Fenster listet alle verfügbaren Tasten und die ihnen aktuell zugeordneten Befehle auf. Erste Spalte: Tastenbezeichnung, Zweite Spalte: Befehlskategorie, Dritte Spalte: Befehl. Wenn eine Taste noch nicht mit einem Befehl belegt ist, wird nur die Tastenbezeichnung angezeigt.

Kategorie: Liste der Befehlskategorien.

Befehl: Liste der Befehle, die zur gerade im Kategorie-Fenster ausgewählten Befehlskategorie gehören.

Tasten: Taste(nkombination), die dem gerade im Befehl-Fenster ausgewählten Befehl zugewiesen ist.

Befehl einer Taste(nkombination) zuweisen, bzw. bestehende Zuweisung ändern:

- Wählen Sie die Taste(nkombination) im Tastaturkürzel-Fenster mit einem Mausklick aus.
- Wählen Sie im Kategorie-Fenster die entsprechende Befehlskategorie und dann im Befehl-Fenster den gewünschten Befehl.
- Drücken Sie Schaltfläche **Ersetzen**. Der gewählte Befehl wird im Tastaturkürzel-Fenster in der Zeile der selektierten Taste(nkombination) eingetragen.

Um ein Tastaturkürzel wieder zu löschen, wählen Sie die entsprechende Zeile im Tastaturkürzel-Fenster an und drücken Schaltfläche **Entfernen**.

12.24 Drucken

Die Befehlskategorie 'Drucken' enthält Befehle zum Erzeugen von Ausdrucken des aktuellen Projekts.

12.24.1 Befehle der Kategorie Drucken

Das Plug-In 'PrintEngine' stellt Befehle der Kategorie 'Drucken' für das Erzeugen einer Druckversion des gesamten Projektinhalts oder ausgewählter Objekte bereit. Es kann direkt zu einem Drucker geschickt oder in einer Datei gespeichert werden..
Standardmäßig sind diese Befehle im Drucken-Menü enthalten. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog verändert werden.

Die Befehle:

- Dokumentieren...
- Seiteneinstellungen...
- Drucken...

12.24.2 Dokumentieren...

Dieser Befehl (Kategorie 'Drucken') öffnet den Dialog **Projekt dokumentieren**, wo eine Dokumentation (Druckversion) des Projekts konfiguriert werden kann.

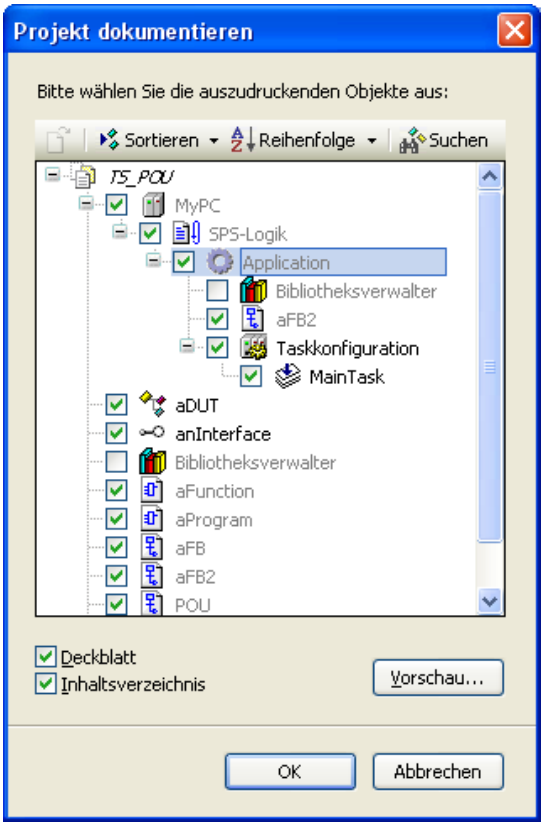


Abb. 236 Dialog 'Projekt dokumentieren'

Die Objekte des Projekts, die im POUs- und Geräte-Fenster definiert sind, werden hier ebenfalls in Baumstruktur dargestellt. Wählen Sie durch Setzen eines Hakens diejenigen Objekte aus, für die eine druckbare Dokumentation erzeugt werden soll. Der Inhalt eines Objekts, das hier grau schattiert dargestellt ist, jst momentan nicht druckbar. Allerdings wird das Objekt, wenn es ausgewählt ist, dennoch als Strukturkomponente in der Dokumentation berücksichtigt und im Inhaltsverzeichnis aufgeführt.

Für das Sortieren und Suchen von Objekten im Dialog stellt die aus anderen View-Fenstern bekannte Werkzeugleiste entsprechende Funktionen bereit.

Wenn die Option **Deckblatt** aktiviert ist, wird die Dokumentation um ein Deckblatt erweitert, dessen Layout in den Seiteneinstellungen , Kategorie Deckblatt definiert wird.

Wenn die Option **Inhaltsverzeichnis** aktiviert ist, wird die Dokumentation um ein Inhaltsverzeichnis erweitert, das die einzelnen Objektkapitel (Objektnamen als Kapitelüberschriften) und die zugehörigen Seitenzahlen darstellt. Die Hierarchie der Kapitel entspricht derjenigen der ausgewählten Objekte im oben dargestellten Strukturbaum. Welche Überschriften-Ebenen im Inhaltsverzeichnis dargestellt werden, wird in den Seiteneinstellungen, Kategorie Dokument, definiert.

Die aktuelle Konfiguration wird als **Standardkonfiguration** für das Projekt gespeichert. Wenn in den aktuellen Einstellungen alle Projektobjekte ausgewählt sind, werden künftig standardmäßig wieder alle Objekte ausgewählt, also auch eventuell neu hinzugefügte. Wenn aktuell einige Objekte aus der Auswahl ausgeschlossen sind, wird standardmäßig genau die konfigurierte Auswahl angeboten, die also neu angelegte Objekte zunächst nicht beinhaltet.

12.24.3 Seiteneinstellungen...

Symbol:

Dieser Befehl (Kategorie 'Drucken') öffnet den Dialog 'Seiteneinstellungen', in dem das Layout für die Seiten einer Druckversion für den Projektinhalt definiert werden kann, welche über die Befehle 'Drucken...' oder 'Dokumentieren...' erzeugt wird.

Die Schaltfläche **Vorschau** kann immer verwendet werden, um die aktuelle Seitenkonfiguration in einer Vorschau darzustellen. Sie öffnet den Dialog 'Seitenansicht', in dem man durch die konfigurierten Seiten blättern kann.

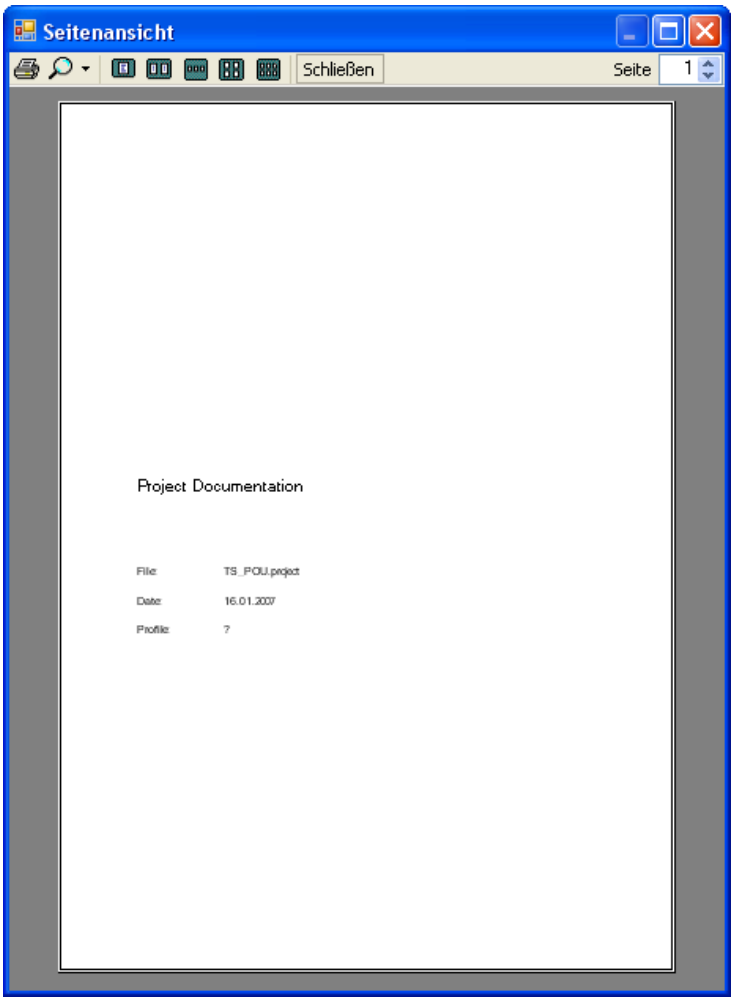


Abb. 237 Dialog 'Seitenansicht' (Vorschau)

Wählen Sie die gewünschte Seitenzahl im Feld **Seite** in der oberen rechten Ecke des Fensters. Der gewünschte **Zoom**-Faktor kann über das Menü gesetzt werden und einer der Ansichts-Modi **Ganze Seite** , **Zwei Seiten**, **Vier Seiten**, **Sechs Seiten** ( ) kann ausgewählt werden. Um die Vorschau auf dem Standard-Drucker auszudrucken verwenden Sie die Schaltfläche . Mit **Schließen** wird das Vorschau-Fenster geschlossen.

Die folgenden Unterdialoge sind auf Registerblättern des Seiteneinstellungen-Dialogs verfügbar:

Papier

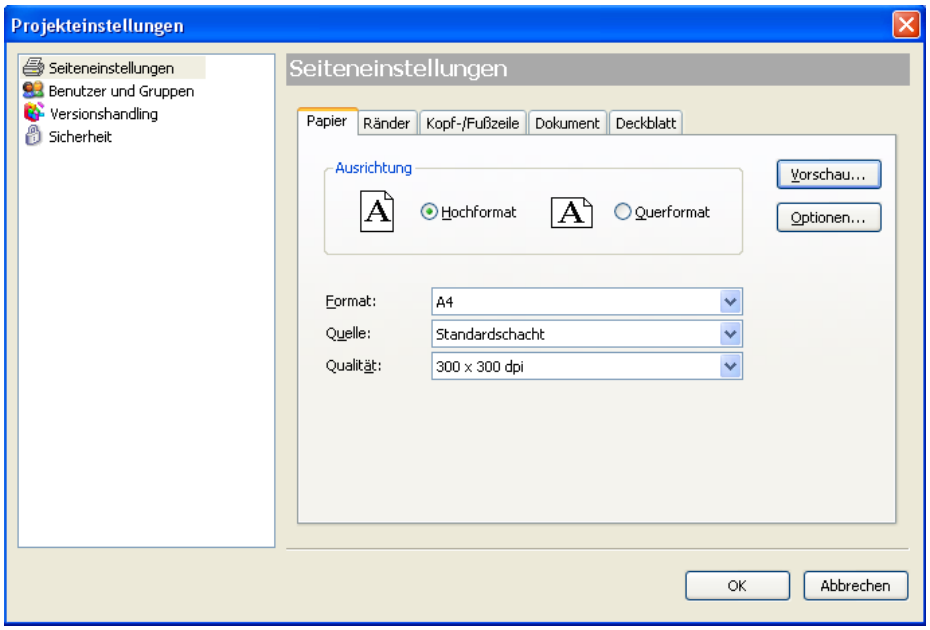


Abb. 238 Dialog 'Seiteneinstellungen', Kategorie 'Papier'

Ausrichtung: Wählen Sie, ob die Seiten der Projekt-Dokumentation Hochformat oder Querformat bekommen sollen.

Format: Wählen Sie das Papierformat, z.B. "A4".

Quelle: Wählen Sie die Quelle für den Papiernachschub am Drucker, z.B. "Standardschacht".

Qualität: Wählen Sie die Druckqualität, z.B. "Draft" für vorläufige Ausdrucke.

Ränder

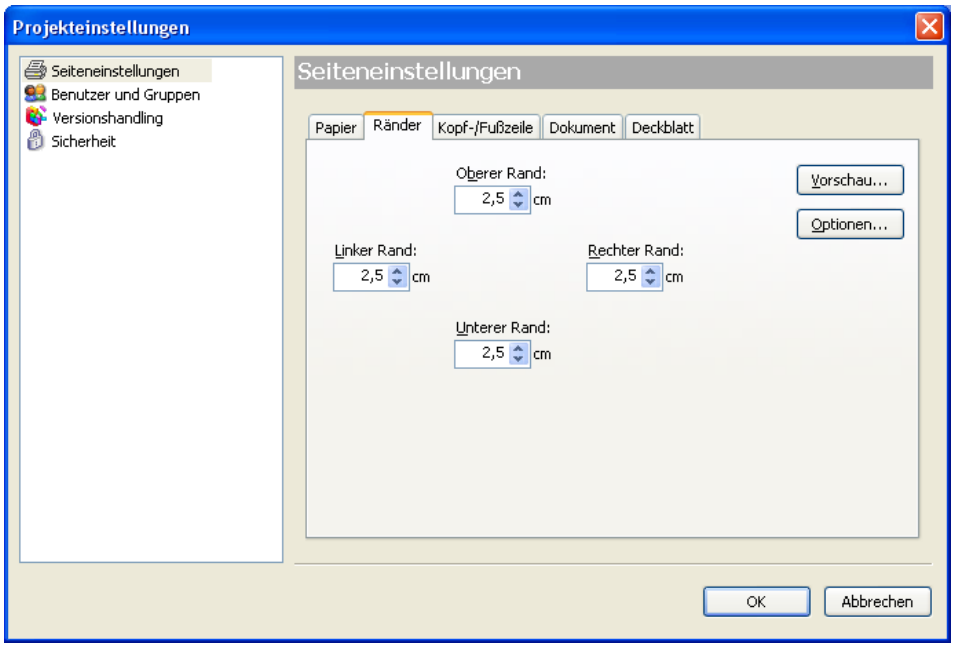


Abb. 239 Dialog 'Seiteneinstellungen', Kategorie 'Ränder'

Definieren Sie hier den oberen, unteren, linken und rechten Rand der Druckseite in Zentimetern. Verwenden Sie die Pfeiltasten, um den Wert in 0,5 Zentimeter Schritten nach oben oder unten zu verändern oder geben Sie den Wert direkt ein.

Kopf-/Fußzeile

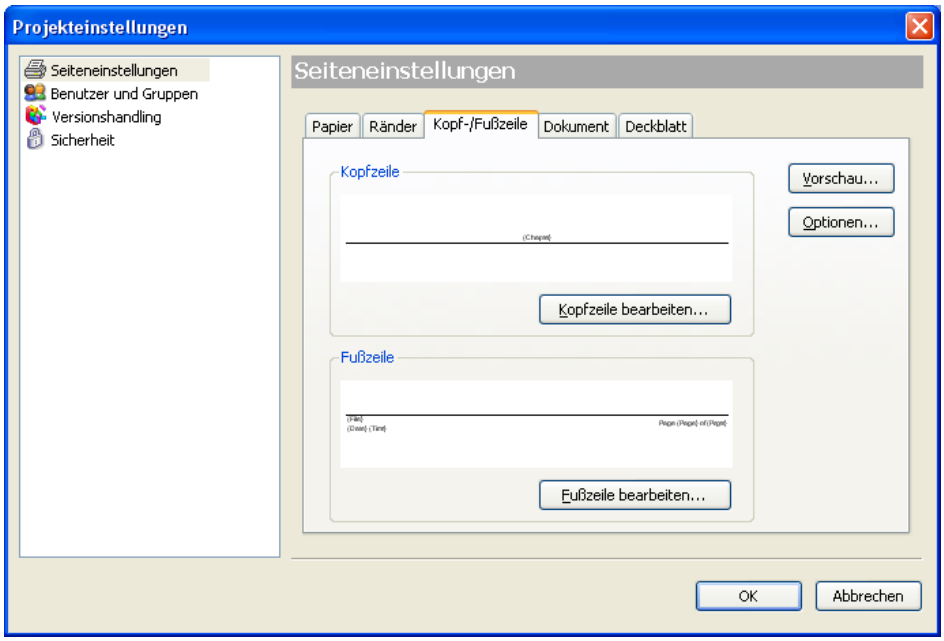


Abb. 240 Dialog 'Seiteneinstellungen', Kategorie 'Kopf-/Fußzeile'

Hier wird die Gestaltung der Kopf- und Fußzeilen vorgenommen. Wählen Sie die Schaltfläche 'Kopfzeile bearbeiten' bzw. 'Fußzeile bearbeiten', um jeweils einen Dialog mit den folgenden Einstellmöglichkeiten zu erhalten:

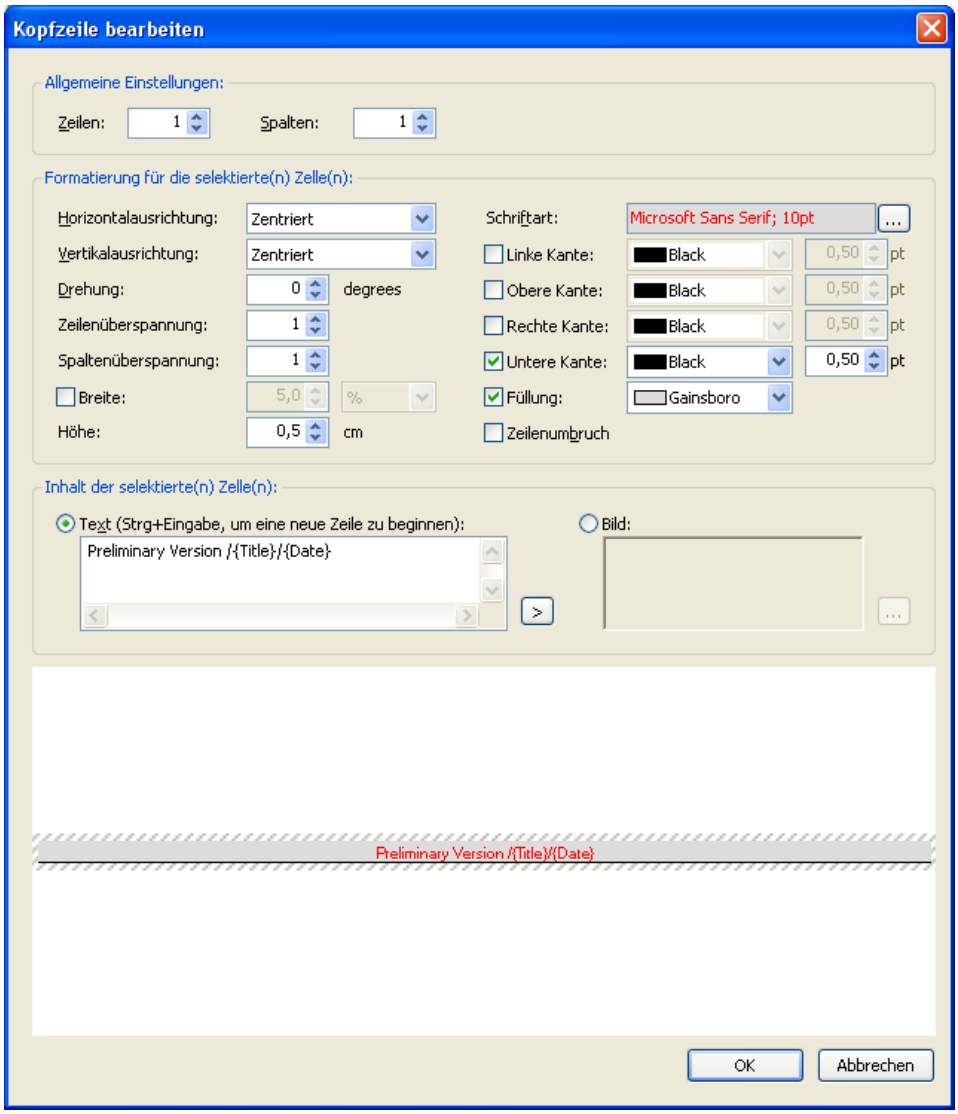


Abb. 241 Dialog 'Kopfzeile bearbeiten' (entspricht 'Fußzeile bearbeiten')

Allgemeine Einstellungen:

Kopf- und Fußzeile sind in Tabellenform strukturiert. Zeilen und Spalten können konfiguriert werden und die entstehenden Zellen können jeweils mit Text oder Bild gefüllt werden. Eine Zelle wird per Mausklick selektiert und daraufhin mit einem grau-gestreiften Rahmen ( ) dargestellt:

Zeilen: Anzahl von Zeilen; wenn die Anzahl erhöht wird, werden die zusätzlichen Zeilen unterhalb der bereits existierenden eingefügt.

Spalten: Anzahl von Spalten; wenn die Anzahl erhöht wird, werden die zusätzlichen Zeilen rechts von den bereits existierenden eingefügt.

Formatierung für die selektierte(n) Zelle(n):

Selektieren Sie die Zelle, die konfiguriert werden soll, per Mausklick im Layout-Feld im unteren Teil des Dialogs.

Horizontalausrichtung: Wählen Sie, ob der Inhalt der Zelle horizontal bündig links oder rechts oder zentriert ausgerichtet sein soll. Standardeinstellung: Zentriert.

Vertikalausrichtung: Wählen Sie, ob der Inhalt der Zelle vertikal oben, unten oder zentriert ausgerichtet sein soll. Standardeinstellung: Zentriert.

Drehung: Wenn der Inhalt der Zelle um ihre Basis gedreht werden soll, kann hier eine Gradzahl für die Drehung angegeben werden. Die "Basis" ist in diesem Fall durch die Horizontalausrichtung definiert (z.B. Horizontalausrichtung = links -> der Text oder das Bild wird um seinen am weitesten links gelegenen und vertikal zentrierten Punkt gedreht). Die Drehung erfolgt immer im Uhrzeigersinn, mögliche Gradangaben sind 0-359. Standardeinstellung: 0.

Zeilenüberspannung: Wenn mehrere Zellen (Zeilen) innerhalb einer Spalte zu einer einzigen verschmolzen werden sollen, kann hier die Anzahl angegeben werden. Beispiel: Eine Angabe von "3" bewirkt, dass die gerade ausgewählte Zelle mit den zwei darunter liegenden verschmolzen wird. Standardeinstellung: 1.

Spaltenüberspannung: Wenn mehrere Zellen (Spalten) innerhalb einer Reihe zu einer einzigen verschmolzen werden sollen, kann hier die Anzahl angegeben werden. Beispiel: Eine Angabe von "3" bewirkt, dass die gerade ausgewählte Zelle mit den zwei rechts davon liegenden verschmolzen wird. Standardeinstellung: 1.

Breite: Wenn diese Option deaktiviert ist, erhalten alle Spalten automatisch die gleiche Breite und die Summe der Spaltenbreiten entspricht der aktuell definierten Seitenbreite. Wenn die Breite einer bestimmten Spalte verändert werden soll, wählen Sie eine beliebige Zelle dieser Spalte aus, aktivieren Sie die Option 'Breite' und geben Sie die gewünschte Spaltenbreite in Zentimetern bzw. in Prozent der gesamten Kopf- oder Fußzeilenbreite ein. Standardeinstellung: deaktiviert.

Höhe: Angabe der Höhe (Zentimeter) der Zeile, in der gerade eine Zelle selektiert ist. Standardeinstellung: 0,5.

Schriftart: Name und Größe des Fonts, der für den Textinhalt der selektierten Zelle definiert ist, werden angezeigt. Zum Ändern der Schriftart-Parameter kann über die Schaltfläche [...] der Standard-Konfigurationsdialog geöffnet werden.

Linke Kante, Obere Kante, Rechte Kante, Untere Kante: Die Sichtbarkeit, Farbe und Liniendicke für Rahmenlinien (Kanten) der selektierten Zelle werden hier definiert. Standardmäßig sind sie schwarz und 0,5 Punkte dick, wobei in der Kopfzeile die untere Kante und in Fußzeilen die obere sichtbar ist.

Füllung: Wenn diese Option aktiviert ist, kann für die selektierte Zelle eine Füllfarbe von der Auswahlliste gewählt werden. Standardmäßig ist eine Zelle nicht gefüllt.

Zeilenumbruch: Wenn diese Option aktiviert ist, wird der Inhalt der selektierten Zelle am rechten Zellenrand umgebrochen, wenn er ansonsten die Zellenbreite überschreiten würde. Standardmäßig ist die Option nicht aktiviert, d.h. der Inhalt wird durch den Zellenrand abgeschnitten.

Inhalt der selektierte(n) Zelle(n):

Die gerade selektierte Zelle kann mit Text oder Bild gefüllt werden. Aktivieren Sie die gewünschte Option. Standardeinstellung: Text.

Text: Im Eingabefeld kann nun ein beliebiger Text eingegeben werden, der in der Zelle erscheinen soll. Mit <Strg>+<Eingabe> kann dabei ein Zeilenumbruch eingefügt werden. Platzhalter können verwendet werden, die dann beim Ausdrucken der Seiten mit den aktuellen Werten ersetzt werden. Die Schaltfläche > öffnet die **Liste der verfügbaren Platzhalter**:

Platzhalter für allgemeine Informationen:

{Page}	Laufende Seitenzahl, z.B. "3"
{Pages}	Gesamtseitenzahl, z.B. "23"
{Date}	Aktuelles Datum dd-mm-yyyy, z.B. "07.07.2006"
{Time}	Aktuelle Uhrzeit hh-mm, z.B. "17:55"

Platzhalter für Informationen, die auf den Projektinformationen basieren:

{Path}	Pfad der Projektdatei, z.B. "D:\projects"
{File}	Name der Projektdatei, z.B. "mach1.project"
{Chapter}	Kapitelnummer + Kapitelüberschrift: Die Kapitelnummer resultiert aus der Position des dokumentierten Objekts im "Dokumentationsbaum", der im Dialog 'Projekt dokumentieren' konfiguriert wurde. Überschriftenebene "1" wird für Objekte verwendet, die im POUs- oder Gerätebaum direkt unterhalb des Root-Knotens platziert sind. Objekte, die darunter eingehängt sind, erhalten die Ebenen 2 und höher. Beispiel: "3.1 POU PLC_PRG.action1" PLC_PRG ist das dritte Objekt auf erster Ebene im POUs-Fenster, das dokumentiert werden soll; action1 ist PLC_PRG zugewiesen, d.h. hängt eingerückt unterhalb von PLC_PRG.
{Size}	Größe der Projektdatei, z.B. "65,42 KB (66.989 bytes)"
{MS-DOS Name}	MS-DOS Name der Projektdatei, z.B. "D:\COD~1.0PR\UN2A48~1.PRO"
{Created}	Erstelldatum der Pojektdatei, z.B. "07.07.2006 17:36"
{Changed}	Datum der letzten Änderung der Projektdatei, z.B.. "07.07.2006 17:36"
{Last Access}	Datum des letzten Zugrifss auf die Projektdatei, z.B. "07.07.2006 17:36"
{Attributes}	Dateiattribute: R=read only / H= Hidden / A=Archive / S=System
{Profile}	Name des aktuellen Profils, z.B. "»PLC Designer« V3.x"
{Company}	Firma (s. Registerblatt 'Zusammenfassung' in den Projektinformationen)
{Title}	Titel des Projekts (s. Registerblatt 'Zusammenfassung' in den Projektinformationen)
{Version}	Version des Projekts (s. Registerblatt 'Zusammenfassung' in den Projektinformationen)
{Default Namespace}	Standard-Namensraum (s. Registerblatt 'Zusammenfassung' in den Projektinformationen); relevant wenn das Projekt als Bibliothek verwendet wird
{Author}	Autor (s. Registerblatt 'Zusammenfassung' in den Projektinformationen)
{Description}	Beschreibung (s. Registerblatt 'Zusammenfassung' in den Projektinformationen)

Bild: Über die Schaltfläche [...] erhält man den Standarddialog für das Auswählen einer Datei. Stellen Sie die gewünschte Bilddatei ein, die daraufhin in der gerade selektierten Zelle angezeigt wird.

Dokument

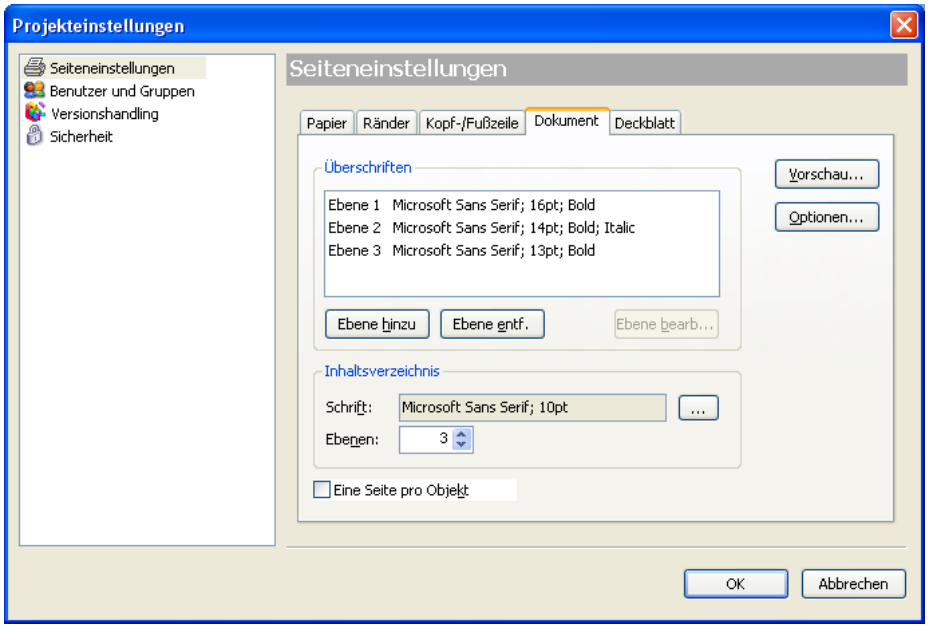


Abb. 242 Dialog 'Seiteneinstellungen', Kategorie 'Dokument'

Hier wird das Format der Kapitelüberschriften definiert und welche Überschriftenebenen im Inhaltsverzeichnis dargestellt werden.

Hinweis: Was ist eine Überschriftenebene? Die Überschriftenebene, die ein Objekt in der Projektdokumentation erhält, wird durch seine Position im POUs- bzw. Gerätebaum. Alle Objekte direkt unterhalb der Root erhalten im Ausdruck Überschriftenebene 1. Objekte, die weiter unten eingehängt sind, erhalten Ebene 2 und höher. Die Nummerierung innerhalb einer Ebene (jeweils durch einen Punkt von der Ebenennummer getrennt) folgt dann der vertikalen Anordnung der Objekte im jeweiligen Ast des Strukturbaums.

Überschriften:

Per Default sind die Standard-Überschriftenebenen 1-3 bereits mit den Standardeinstellungen für die **Schrift** konfiguriert (wie in der oben stehenden Abbildung zu sehen). Sie werden in der **Überschriften**-Liste aufgeführt.

Weitere Ebenen, die ev. in der Projektdokumentation verwendet werden, werden automatisch so dargestellt wie es für den untersten der Überschriften-Liste definiert ist (z.B. wird im Standardfall Ebene 4 genauso aussehen wie Ebene 3). Wenn Sie die Darstellung einer dieser tieferen Ebenen jedoch explizit konfigurieren wollen, müssen Sie sie vorher mit Hilfe von über Schaltfläche **Ebene hinzu** zur Überschriftenliste hinzufügen. Denn nur Überschritenebenen, die in der Liste stehen, können bezüglich ihrer Schriftart bearbeitet werden. Dazu wird der Listeneintrag selektiert und über Schaltfläche **Ebene bearbeiten....** der Standarddialog zur Schriftart-Konfiguration geöffnet und die entsprechenden Einstellungen vorgenommen..

Jeweils der unterste Eintrag in der Überschriftenliste kann mit Hilfe von Schaltfläche **Ebene entfernen** gelöscht werden.

Inhaltsverzeichnis:

Hier wird definiert, welche **Schrift** (Schaltfläche ... öffnet den Dialog zur Schriftart-Konfiguration) und wieviele Überschriften**Ebenen** (gewünschte Zahl eingeben) im Inhaltsverzeichnis der Projektdokumentation dargestellt werden sollen..

Eine Seite pro Objekt: Wenn diese Option aktiviert ist, wird jedes Objekt auf einer separaten Seite dokumentiert. Standardmäßig werden diesbezüglich keine Seitenumbrüche eingefügt.

Deckblatt

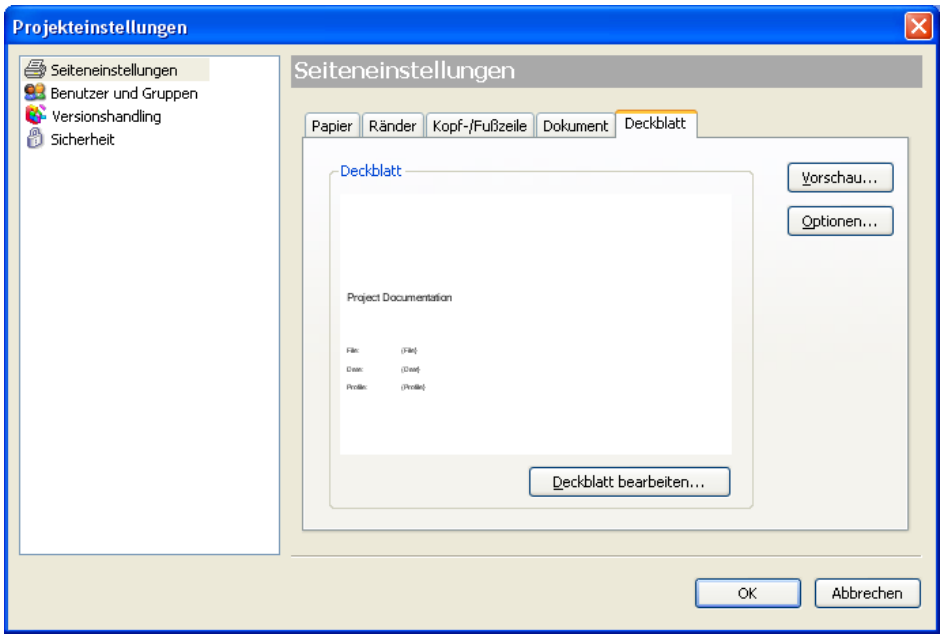


Abb. 243 Dialog Seiteneinstellungen, Kategorie 'Deckblatt'

Hier wird das Layout des Deckblatts für die Projektdokumentation festgelegt. Das Deckblatt wird nur hinzugefügt, wenn die Option 'Deckblatt' im Dialog 'Projekt dokumentieren' aktiviert ist, wo auch die Inhalte des Dokumentations-Sets definiert werden.

Über die Schaltfläche **Deckblatt bearbeiten** gelangt man zum Konfigurationsdialog, der die gleichen Einstellmöglichkeiten bietet wie die Dialoge Kopfzeile bzw. Fußzeile bearbeiten.

12.24.4 Drucken...

Symbol:

Dieser Befehl (Kategorie 'Drucken') dient dazu, den Inhalt des aktiven Editorfensters auf einem Drucker bzw. in einer Datei auszugeben.

Dazu öffnet der Standarddialog zum Einstellen des Druckbereichs , der Druckerparameter und der Anzahl der Kopien. Wenn die Option 'Ausgabe in Datei' aktiviert ist, kann anstelle der Ausgabe auf einem Drucker eine entsprechende Datei definiert werden.

Das Layout für die zu druckenden Seiten wird in den Seiteneinstellungen.... festgelegt.

Wenn der Dialog mit OK geschlossen wird, während bereits ein Druckauftrag läuft, stoppt dieser nach der nächsten Seite.

12.25 Rezepturverwalter

Die Befehlskategorie 'Rezepturverwalter' enthält Befehle zum Anlegen von Rezepturen im Rezepturverwalter.

12.25.1 Rezepturverwalter Befehle

Das Plug-In 'RecipeManagement Editor' stellt Befehle der Kategorie 'Rezepturverwalter' für das Arbeiten im Rezepturverwalter-Editor bereit.

Diese Befehle sind im Kontextmenü verfügbar, wenn eine Rezepturdefinition im Editor bearbeitet wird. Standardmäßig stehen sie dann auch in einem Menü '**Rezept**" zur Verfügung. Gegebenenfalls kann die Menüstruktur mit Hilfe des Anpassen-Dialogs verändert werden.

Die Befehle:

- Rezeptur einfügen
- Rezeptur entfernen

12.25.2 Rezeptur einfügen

Symbol:

Dieser Befehl (Kategorie 'Rezepturverwalter') fügt eine neue Spalte für eine Rezeptur in der gerade bearbeiteten Rezepturdefinition ein. Der Dialog **Neue Rezeptur** wird geöffnet, wo ein Name für die Rezeptur eingegeben werden muss. Wenn die Werte einer existierenden Rezeptur kopiert werden sollen, wählen Sie die entsprechende aus der Auswahlliste.

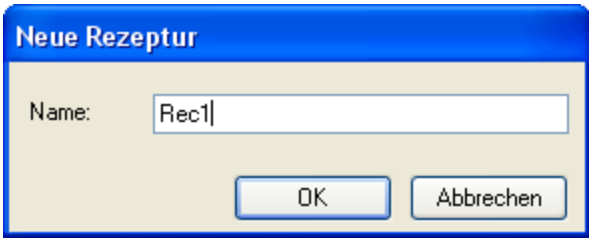


Abb. 244 Dialog 'Neue Rezeptur'

Hinweis: Eine neue Rezeptur kann auch im Online-Modus über ein entsprechend konfiguriertes Visualisierungselement (Eingabekonfiguration, Kategorie 'Internes Kommando') angelegt werden.

12.25.3 Rezeptur entfernen

Symbol:

Dieser Befehl (Kategorie 'Rezepturverwalter') löscht eine Rezeptur aus der gerade geöffneten Rezepturdefinition.

Wählen Sie eines der Felder in der Rezepturspalte und führen den Befehl aus. Die Spalte wird entfernt.

Hinweis: Rezepturen können auch im Online-Modus über ein entsprechend konfiguriertes Visualisierungselement (Eingabekonfiguration, Kategorie 'Internes Kommando') gelöscht werden.

12.26 Intelligente Code-Bearbeitung

Die Befehlskategorie 'Intelligente Code-Bearbeitung' enthält Funktionen, die die Eingabe von Programm-Code erleichtern, wie z.B. Eingabehilfe- und Autodeklarations-Dialoge.

12.26.1 Eingabehilfe

Befehle der Kategorie Intelligente Code-Bearbeitung - Eingabehilfe

Das Plug-In 'Input Assistant' stellt Befehle für die Befehlskategorie 'Intelligente Code-Bearbeitung' bereit. Derzeit handelt es sich um den Befehl zum Öffnen des Eingabehifle-Dialogs.

Standardmäßig ist der Befehl im Menü 'Bearbeiten' zu finden. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen- Dialogs umkonfiguriert werden.

Eingabehilfe...

Symbol:

Der Eingabehilfe-Dilaog bzw. der Befehl 'Eingabehilfe' (Befehlskategorie Intelligente Code-Bearbeitung) sind nur verfügbar, wenn der Cursor gerade in einem Texteditor-Fenster steht. Der Dialog bietet alle Programmierelemente an, die an der aktuellen Cursor-Position eingefügt werden können.

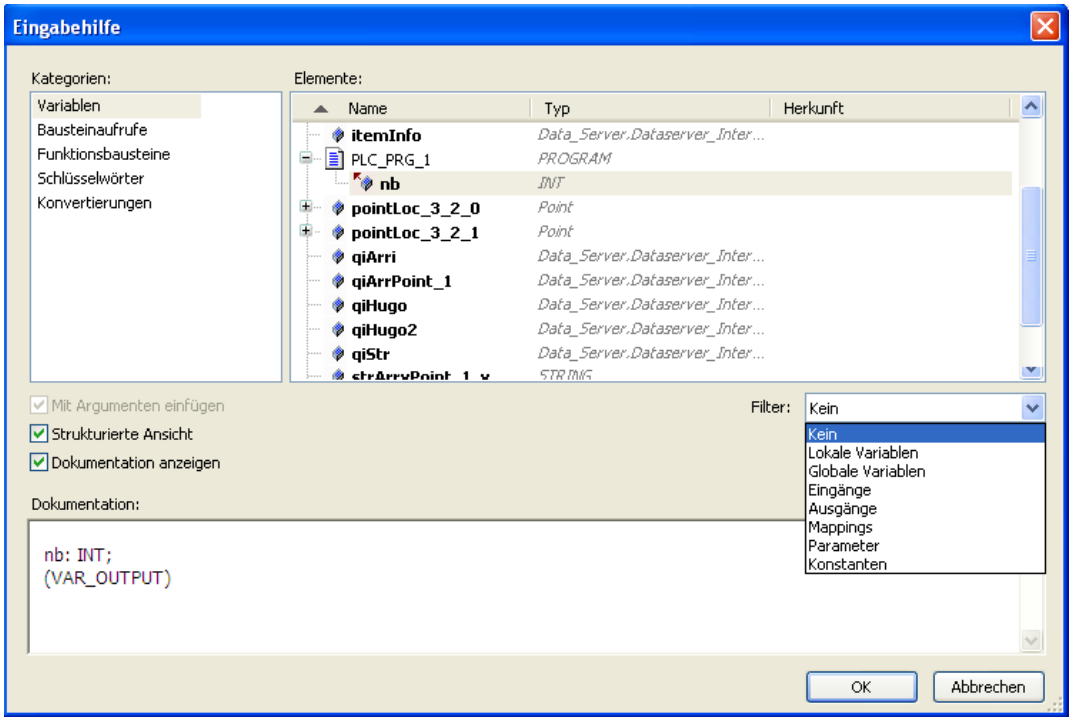


Abb. 245 Eingabehilfe-Dialog

Die Elemente sind nach **Kategorien** sortiert. Für die Kategorie "Variablen" kann zusätzlich ein Filter für den Variablentyp gesetzt werden, wie "Lokale Variablen", "Globale Variablen", "Konstanten" etc.

Für die gerade ausgewählte Kategorie werden die verfügbaren **Elemente** und - abhängig von der Kategorie - auch ihr Datentyp und Ursprung im Elemente-Fenster angezeigt. Der Ursprung wird für E/A-Variablen (Pfad innerhalb des Gerätebaums) und bibliotheksdefinierte Variablen (Bibliotheksname und -kategorie) angegeben. Wenn Option Strukturierte Darstellung aktiviert ist, werden die Elemente zusätzlich mit Icons versehen in einem Strukturbaum dargestellt, ansonsten werden sie 'flach' angeordnet, wobei allerdings der Baustein, zu dem sie gehören angegeben wird (z.B. 'GVL1.gvar1'). Die Elemente können nach Name, Datentyp oder Ursprung aufsteigend oder absteigend alphabetisch sortiert werden, indem mit der Maus in den entsprechenden Spaltentitel geklickt wird (Pfeil-Symbol nach oben bzw. unten).

Hinweis: Wenn Objekte gleichen Namens und gleicher Kategorie sowohl global (POUs Fenster) als auch einer Applikation zugeordnet (Gerätebaum) angelegt sind, erscheint nur 1 Eintrag in der Eingabehilfe, da die Verwendung der üblichen Aufrufpriorität (applikationszugeordnet vor global) folgt.

Wenn die Option **Dokumentation** anzeigen aktiviert ist, wird der Dialog um das Feld 'Dokumentation' erweitert. Wenn das ausgewählte Element eine Variable ist, der eine Adresse zugewiesen bzw. bei deren Deklaration ein Kommentar angegeben ist, werden diese hier angezeigt.

Mit Argumenten einfügen: Wenn diese Option aktiviert ist, werden Elemente, die Argumente besitzen, wie z.B. Funktionen, mit diesen Argumenten an der Cursor-Position eingefügt. Beispiel: Wenn Funktionsblock fb1, der eine Eingabevariable fb1_in und eine Ausgabevariable fb1_out enthält, "mit Argumenten" eingefügt wird, sieht dies im Editor folgendermaßen aus: fb1(fb1_in:= , fb1_out=>)

Variable deklarieren

Befehle der Kategorie Intelligente Code-Bearbeitung - Variable deklarieren

Das 'AutoDeclare' Plug-In liefert einen Befehl Variable deklarieren... für die Befehlskategorie 'Intelligente Code-Bearbeitung'.

Standardmäßig ist der Befehl im Bearbeiten-Menü enthalten. Wenn erforderlich, verwenden Sie den Anpassen-Dialog um die Menükonfiguration zu verändern. Beachten Sie, dass für die Funktionalität 'Intelligentes Kodieren' auch bestimmte Einstellungen im Optionen-Dialog gemacht werden können.

Variable deklarieren...

Standard Tastaturkürzel: <Umsch> + <F2>

Dieser Befehl (Kategorie 'Intelligente Code-Bearbeitung') öffnet den Dialog 'Variable deklarieren', der die Deklaration einer Variablen unterstützt. Dazu muss der Cursor in einer Zeile des Implementierungsteils des Editors stehen, die eine noch nicht deklarierte Variable enthält, oder es muss eine bereits deklarierte Variable selektiert sein. Wenn das Öffnen des Dialogs automatisch erfolgen soll, sobald eine Zeile verlassen wird, die eine noch nicht deklarierte Variable enthält, muss die "Autodeclare"-Funktion (Autodeklaration) in den Optionen für Intelligente Code-Bearbeitung aktiviert sein.

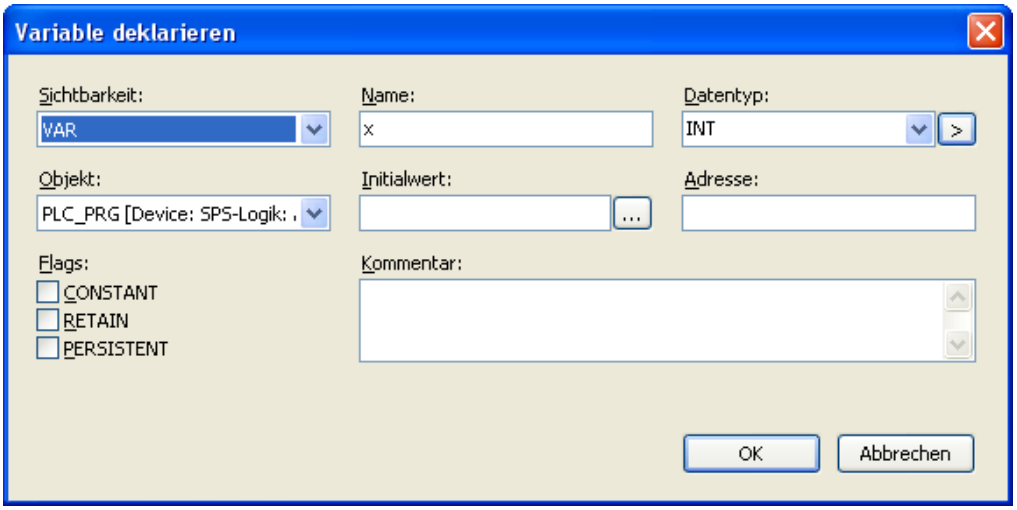


Abb. 246 Dialog für Variablendeklaration

Einige Felder im Dialog werden automatisch mit Standardwerten gefüllt, können aber bearbeitet werden:

Name: Standardmäßig steht hier der Name der noch nicht deklarierten Variable, wie er im Editor eingegeben wurde.

Objekt: Standardmäßig steht hier der Name des Projektobjekts, das gerade bearbeitet wird. Um ein anderes Objekt anzugeben, in dem die Variable deklariert werden soll, wählen Sie eines aus der Auswahlliste. Diese enthält beispielsweise, wenn Sie gerade eine globale Variable deklarieren (Sichtbarkeit: VAR_GLOBAL), alle verfügbaren globalen Variablenlisten des Projekts.

Datentyp: Standardmäßig steht hier INT, wenn es sich gerade um die erste zu deklarierende Variable in der Editorzeile handelt. Wenn es allerdings bereits eine deklarierte Variable in der Zeile gibt, wird der Datentyp dieser Variable hier vorgegeben. Um die Einstellung zu ändern, können Sie über die Schaltfläche > die Eingabehilfe in Anspruch nehmen, die alle möglichen Datentypen anbietet. Wenn eine Array-Variable deklariert werden soll, kann der Array-Assistent verwendet werden, der ebenfalls über die Pfeil-Schaltfläche erreichbar ist. Siehe unten eine Beschreibung.

Sichtbarkeit: Standardmäßig steht hier VAR für "lokale Variable", alternativ kann aus der Auswahlliste ein anderer Gültigkeitsbereich eingestellt werden.

Initialwert: Standardmäßig leer. Ein dem Datentyp der Variablen entsprechender, gültiger Wert, mit dem die Variable initialisiert werden soll, kann hier eingegeben werden. Wenn das Feld leer bleibt, wird sie mit dem Standardwert initialisiert. Noch nicht implementiert: Press button ... to get assistance by.....

Adresse: Standardmäßig leer. Die Variable kann hier direkt einer IEC-Adresse zugewiesen werden ("AT"-Deklaration). Beispiel: Wenn für Variable "b" vom Datentyp "INT" die Adresse "%IB0" eingetragen wird, ergibt sich folgende Deklarationszeile: "b AT %IB0 : INT;"

Kommentar: Standardmäßig leer. Hier kann Text eingegeben werden, der der Deklaration als Kommentar hinzugefügt wird. Umbrüche können durch die Tastenkombination <Strg> + <Eingabe> eingefügt werden. Der Kommentar wird dann im Deklarationseditor in der Zeile über der Variablendeklaration angezeigt.

Flags: CONSTANT, **RETAIN**, **PERSISTENT**: Standardmäßig nichts aktiviert. Sie können durch Aktivieren des entsprechenden "flags" definieren, dass es sich bei der Variablen um eine Konstante oder eine remanente Variable handelt. Das entsprechende Attribut-Schlüsselwort wird dann in der Deklaration hinter dem in "Sichtbarkeit" definierten Schlüsselwort eingefügt, z.B. "VAR CONSTANT". Die PERSISTENT Option ist nur verfügbar, wenn es eine Persistente Variablen Liste gibt.

Autodeklaration von Arrays:

Wenn Sie den Assistenten für die Deklaration von Array-Variablen verwenden möchten, wählen Sie über die Pfeiltaste > hinter dem Datentyp-Feld den Befehl **Array-Assistent**. Der Dialog **Array** wird geöffnet:

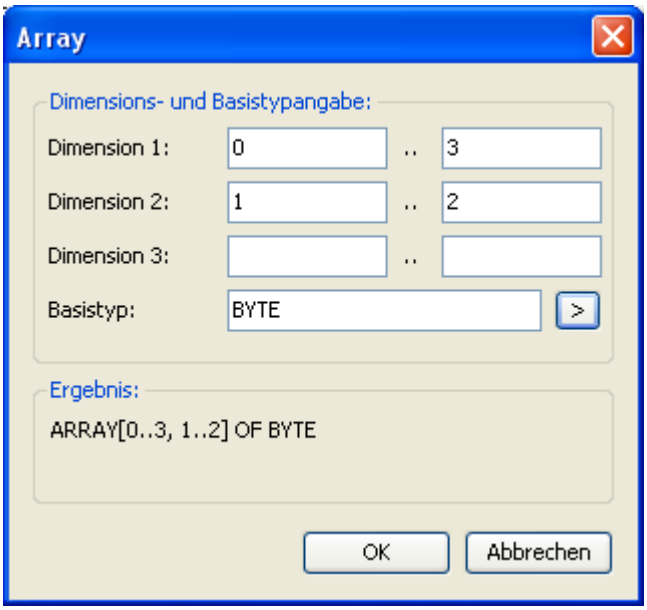


Abb. 247 Array Dialog, Beispiel

Mindestens die mit einem Ausrufezeichen versehenen Felder müssen ausgefüllt werden. Definieren Sie die Feldgrößen (**Dimension)** durch Eingeben der unteren und oberen Grenzen, und den **Basistyp** der Variablen, wobei die Pfeil-Schaltfläche zur Eingabehilfe führt.

Im unteren Teil des Dialogs (Ergebnis) wird eine Vorschau der aktuell definierten Deklaration angezeigt.

Siehe auch die Hilfeseite zu Arrays.

Mit **OK** wird der Deklarationsdialog geschlossen und die Variablendeklaration erscheint im Deklarationseditor gemäß IEC-Syntax.

12.27 AS

Die Befehlskategorie 'AS' enthält die Befehle zum Arbeiten im Ablaufsprachen (AS) Editor.

12.27.1 AS-Editor Befehle

Die Befehle der Kategorie 'AS' werden mit dem Plug-In 'SFC Editor' installiert und stehen für die Programmierung im Ablaufsprachen-Editor (AS-Editor) zur Verfügung .

Standardmäßig sind sie im "AS"-Menü enthalten, das in der Menüleiste erscheint, wenn der AS-Editor aktiv ist. Wenn nötig, können Sie den Anpassen-Dialog, Kategorie 'Menü', verwenden um die Menükonfiguration zu verändern.

Die Befehle:

- Initialschritt
- Schritt-Transition einfügen
- Schritt-Transition danach einfügen
- Parallel
- Alternativ
- Verzweigung einfügen
- Verzweigung rechts einfügen
- Aktionsassoziation einfügen
- Aktionsassoziation danach einfügen
- Sprung einfügen
- Sprung danach einfügen
- Makro einfügen
- Makro danach einfügen
- Makro anzeigen
- Makro verlassen

12.27.2 Initialschritt

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um den gerade selektierten Schritt zum Initialschritt zu machen.

Nach Ausführen des Befehls verändert sich der Rahmen des Schritt-Elements in eine Doppellinie. Der Schritt, der zuvor Initialschritt war, wird automatisch zu einem "normalen" Schritt und mit einfachem Rahmen dargestellt.

Der Befehl kann nützlich sein, wenn das Diagramm umgestellt werden soll.

Wenn ein neues AS-Objekt angelegt wird, enthält es automatisch einen Initialschritt, gefolgt von einer Transition (TRUE) und einem Sprung zurück zum Initialschritt.

Die Eigenschaft "Initialschritt" kann auch im Eigenschaften-Dialog eines Schritt-Elementes aktiviert oder deaktiviert werden, dabei erfolgt allerdings keine automatische Anpassung der Einstellung bei anderen Schritten.

Beachten Sie die Möglichkeit, im Online-Betrieb das Diagramm mit Hilfe der Variablen SFCInit bzw. SFCReset auf den Initialschritt zurückzusetzen.

12.27.3 Schritt-Transition einfügen

Symbol:

Batch-Befehl: file open

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um einen Schritt und eine Transition **vor** der gerade angewählten Position einzufügen.

Die Anordnung (Reihenfolge) der neuen Schritt-Transition-Kombination hängt davon ab, ob beim Einfügen gerade ein Schritt oder eine Transition selektiert ist. Die grundsätzliche Reihenfolge Schritt-Transition-Schritt-Transition-.... wird automatisch eingehalten. Sehen Sie in der folgenden Abbildung ein Beispiel:

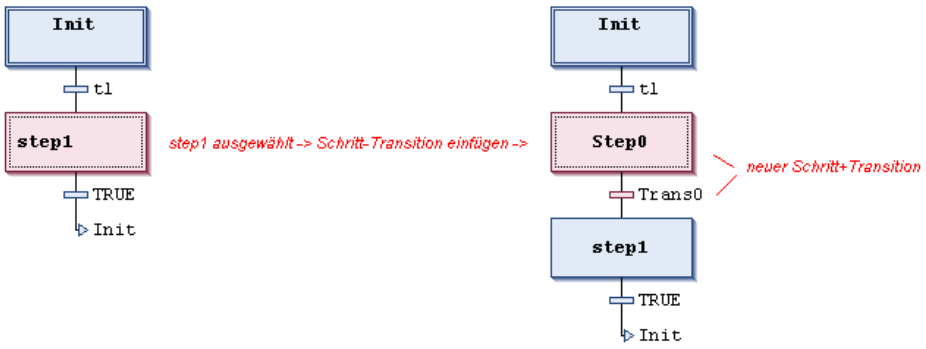


Abb. 248 Schritt-Transition eingefügt

Der neue Schritt wird per Default "Step<n>". n genannt. n ist eine fortlaufende Zahl, beginnend mit "0" für den ersten Schritt, der zusätzlich zum Initialschritt eingefügt wird.

Die neue Transtion wird entsprechend per Default "Trans<n>" genannt.

Die Default-Namen können direkt editiert werden (Mausklick auf Text öffnet Eingabefeld).

12.27.4 Schritt-Transition danach einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um einen Schritt und eine Transition **nach** der gerade angewählten Position einzufügen.

Die Anordnung (Reihenfolge) der neuen Schritt-Transition-Kombination hängt davon ab, ob beim Einfügen gerade ein Schritt oder eine Transition selektiert ist. Die grundsätzliche Reihenfolge Schritt-Transition-Schritt-Transition-.... wird automatisch eingehalten. Sehen Sie in der folgenden Abbildung ein Beispiel:

In diesem Beispiel wird die neue Schritt-Transition-Kombination nach Transition "TRUE" eingefügt, die vor Ausführen des Befehls ausgewählt wurde.

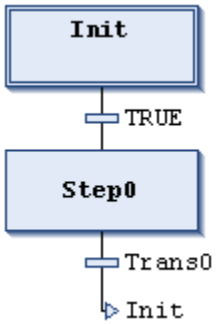


Abb. 249 Schritt-Transition "danach" eingefügt, Beispiel

Der neue Schritt wird per Default "Step<n>". n genannt. n ist eine fortlaufende Zahl, beginnend mit "0" für den ersten Schritt, der zusätzlich zum Initialschritt eingefügt wird.

Die neue Transtion wird entsprechend per Default "Trans<n>" genannt.

Die Default-Namen können direkt editiert werden (Mausklick auf Text öffnet Eingabefeld).

12.27.5 Parallel

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um die gerade selektierte Alternative Verzweigung in eine Parallele Verzweigung umzuwandeln.

Beachten Sie, dass nach der Umwandlung einer Verzweigung die Anordnung der Schritte und Transitionen vor und nach der Verzweigung geprüft und angepaßt werden muss.

Siehe auch: Alternativ

12.27.6 Alternativ

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um die gerade selektierte Parallele Verzweigung in eine Alternative Verzweigung umzuwandeln.

Beachten Sie, dass nach der Umwandlung einer Verzweigung die Anordnung der Schritte und Transitionen vor und nach der Verzweigung geprüft und angepaßt werden muss.

Siehe auch: Parallel

12.27.7 Verzweigung einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um eine Verzweigung **links** von der aktuellen Position einzufügen.

Siehe ansonsten die Beschreibung für Verzweigung rechts einfügen.

12.27.8 Verzweigung rechts einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um eine Verzweigung **rechts** von der aktuellen Position einzufügen.. (Um sie jeweils links einzufügen, verwenden Sie Befehl Verzweigung einfügen)

- Wenn das oberste Element der grade ausgewählten Elemente eine Transition oder eine Alternative Verzweigung ist, wird eine Alternative Verzweigung eingefügt.
- Wenn das oberste Element der gerade ausgewählten Elemente ein Schritt, ein Makro, ein Sprung oder eine Parallele Verzweigung ist, wird eine Parallele Verzweigung mit **Sprungmarke** "Branch<x>" eingefügt, wobei x eine fortlaufende Zahl ist, die für die erste Marke im Diagramm mit "0" startet. Dieser Default-Name der Sprungmarke kann editiert werden. Die Sprungmarke kann bei einem Sprung als Ziel angegeben werden.
- Wenn gerade ein gemeinsames Element einer bestehenden Verzweigung selektiert ist (horizontale Linie), wird die neue Verzweigung ganz rechts als weiterer Arm hinzugefügt. Wenn gerade ein gesamter Arm einer bestehenden Verzweigung selektiert ist, wird die neue Verzweigung direkt rechts davon als neuer Arm hinzugefügt

Hinweis: Beachten Sie dass eine Verzweigung mit Befehl 'Alternativ' bzw 'Parallel' in den jeweils anderen Typ übergeführt werden kann.

Beispiele:

Parallele Verzweigung:

In der folgenden Abbildung sehen Sie eine neu eingefügte Parallele Verzweigung, erzeugt mit Befehl 'Verzweigung rechts einfügen' während step11 ausgewählt war. Automatisch wird ein Schritt ("Step2" im Beispiel) eingefügt.

Abarbeitung im Online-Modus: Wenn t2 TRUE liefert, wird Step2 sofort nach step11 ausgeführt, bevor t3 ausgewertet wird.

Es werde also im Gegensatz zu Alternativen Verzweigungen beide Zweige ausgeführt.

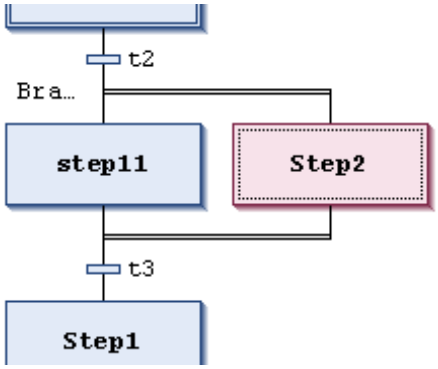


Abb. 250 Parallele Verzweigung

Alternative Verzweigung:

In der folgenden Abbildung sehen Sie eine neu eingefügte Alternative Verzweigung, erzeugt mit Befehl 'Verzweigung rechts einfügen' während Transition t4 ausgewählt war. Automatisch wird ein Schritt ("Step32" im Beispiel), eine vorausgehende und eine nachfolgende Transition (t41, t42) eingefügt.

Abarbeitung im Online-Modus: Wenn Step3 aktiv ist, werden die nachfolgenden Transitionen (t4, t41) von links nach rechts ausgewertet. Der erste Arm der Verzweigung, in dem die erste Transition TRUE liefert, wird ausgeführt. Somit wird im Gegensatz zu einer Parallelen Verzweigung nur ein Arm ausgeführt.

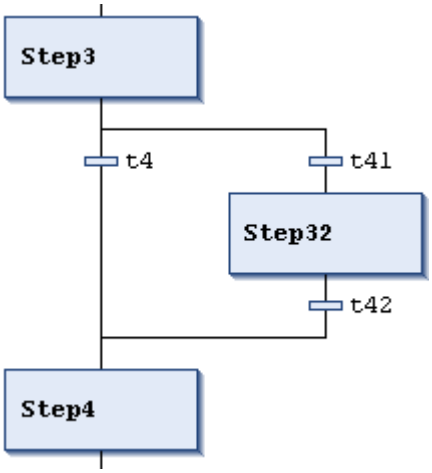


Abb. 251 Alternative Verzweigung

12.27.9 Aktionsassoziation einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um einem Schritt eine IEC- Aktion zuzuweisen.

Das Aktion-Element wird rechts neben dem gerade ausgewählten Schritt-Element eingefügt.

Wenn dem Schritt bereits eine oder mehrere Aktionen zugewiesen sind, werden diese in einer "Aktionenliste" untereinander dargestellt. Die neue Aktion wird dann folgendermaßen platziert:

- Wenn das Schritt-Element selektiert ist, als erste Aktion des Schrittes, d.h. an oberster Position der Aktionenliste.
- Wenn eine der vorhandenen Aktionen in der Aktionenliste des Schrittes selektiert ist, direkt vor, d.h. oberhalb von dieser.

Siehe auch Aktionsassoziation danach einfügen.

Der linke Teil des Aktion-Elements enthält den **Qualifizierer**, standardmäßig "N", im rechten Teil muss der **Aktionsname** eingegeben werden. Dazu klicken Sie in die Box, um einen Editierrahmen zu erhalten. Eine Aktion mit dem eingegebenen Namen muss im Projekt als POU angelegt sein. Siehe hierzu Befehl 'Objekt hinzufügen'.

Der Qualifizierer kann ebenfalls editiert werden.

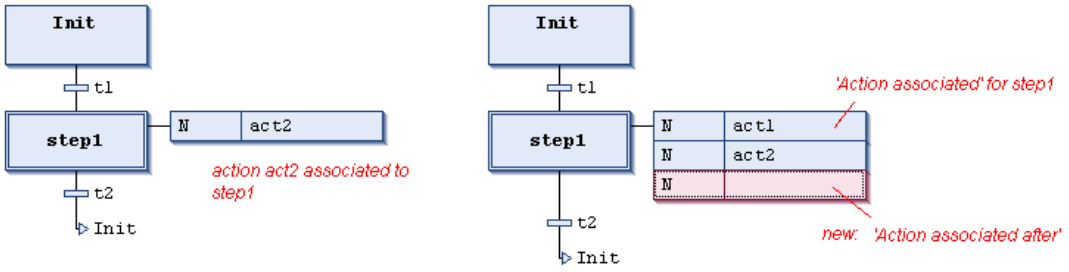


Abb. 252 Einem Schritt zugewiesene Aktionen, Beispiele

12.27.10 Aktionsassoziation danach einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um einem Schritt eine weitere IEC- Aktion zuzuweisen.

Sehen Sie hierzu bitte die Beschreibung für Befehl Aktionsassoziation einfügen. Der Unterschied besteht darin, dass die neue Aktion mit "...danach einfügen" nicht an der ersten, sondern letzten Position der Aktionenliste, bzw. nicht oberhalb, sondern unterhalb der gerade in dieser Liste ausgewählten Aktion platziert wird.

12.27.11 Sprung einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um ein Sprung-Element **vor**dem gerade ausgewählten Element einzufügen.

Der Sprung wird automatisch mit Sprungziel "Step" eingefügt. Ersetzen Sie "Step" durch den Namen eines Schrittes oder der Sprungmarke einer Parallelen Verzweigung, zu dem/der gesprungen werden soll.

Siehe auch: Sprung danach einfügen

12.27.12 Sprung danach einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um ein Sprung-Element **nach** dem gerade ausgewählten Element einzufügen.

Der Sprung wird automatisch mit Sprungziel "Step" eingefügt. Ersetzen Sie "Step" durch den Namen eines Schrittes oder der Sprungmarke einer Parallelen Verzweigung, zu dem/der gesprungen werden soll.

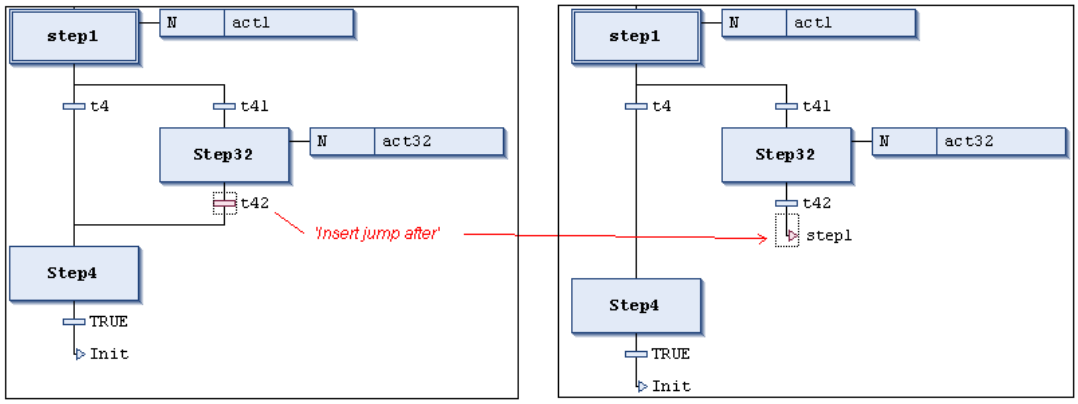


Abb. 253 Beispiel, Sprung danach einfügen

Siehe auch: Sprung einfügen

12.27.13 Makro einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um ein Makro-Element **vor** dem gerade ausgewählten Element einzufügen. Standardmäßig wird der Makroname "Macro<x>" vergeben, wobei x eine fortlaufende Nummer ist, die mit "0" beim ersten eingefügten Makro beginnt. Dieser Name kann editiert werden.

Um ein Makro zu bearbeiten oder darzustellen, kann mit dem Befehl Makro anzeigen der Makro-Editor geöffnet werden.

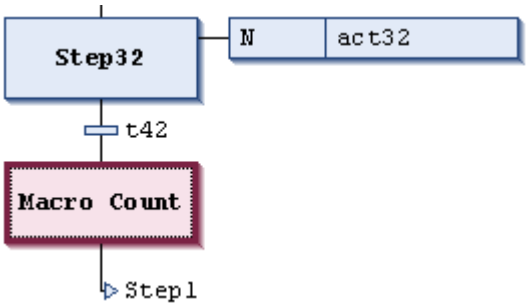


Abb. 254 Beispiel, Makro (gerade ausgewählt) in einem AS-Diagramm

Siehe auch: Makro danach einfügen

12.27.14 Makro danach einfügen

Symbol:

Dieser Befehl (Kategorie 'AS') wird im AS-Editor verwendet, um ein Makro nach dem gerade ausgewählten Element einzufügen. Siehe entsprechende Informationen bei Makro einfügen.

12.27.15 Makro anzeigen

Symbol:

Dieser Befehl (Kategorie 'AS') ist im AS-Editor verfügbar, wenn ein Makro-Element im AS-Diagramm ausgewählt ist. Er dient dazu, das Makro im **Makro-Editor** zur Ansicht oder Bearbeitung zu öffnen.

Die Hauptansicht des AS-Editors wird geschlossen und anstelle dessen der Makro-Editor geöffnet. Es handelt sich ebenfalls um einen AS-Editor, wo Sie nun den Teil des AS-Diagramms ansehen und bearbeiten können, der in der Hauptansicht nur als Makro-Box dargestellt wird. Ein Zoom-Menü ist auch hier in der unteren rechten Ecke verfügbar.

Um zur Hauptansicht des AS-Editors zurückzukehren, verwenden Sie den Befehl Makro verlassen.

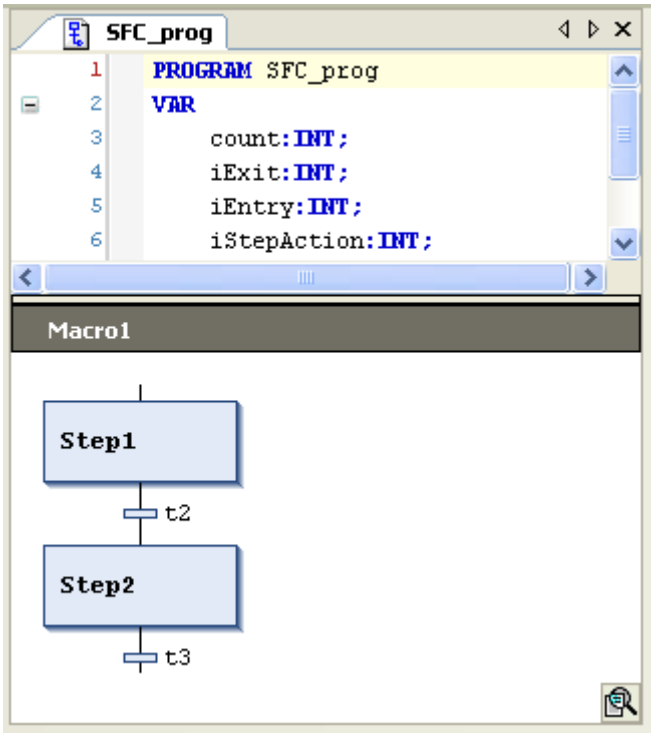


Abb. 255 Makro-Editor, Beispiel

12.27.16 Makro verlassen

Symbol:

Dieser Befehl (Kategorie 'AS') ist im AS-Editor verfügbar, wenn gerade der Makro-Editor geöffnet ist (über Befehl Makro anzeigen).

Der Makro-Editor wird wieder geschlossen und man kehrt zur Hauptansicht des AS-Editors zurück. Der Befehl kann im Offline- und Online-Modus verwendet werden.

12.28 Quellcodeverwaltung

Die Befehlskategorie 'Quellcodeverwaltung' enthält Befehle für das Verwalten eines Projekts unter Quellcodeverwaltung, d.h. über den ENI Server in einer externen Datenbank .

12.28.1 Befehle der Kategorie Quellcodeverwaltung

Die Befehle der Kategorie 'Quellcodeverwaltung' werden mit dem Plug-In 'Source Control' bereit gestellt und dienen der Handhabung von Objekten eines Projekts, die über ENI (Engineering Interface) in einer externen Projektdatenbank verwaltet werden.

Standardmäßig sind die Befehle im Projekt-Menü in einem Untermenü 'Quellcodeverwaltung' zu finden. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog umkonfiguriert werden.

Beachten Sie, dass als Voraussetzung für das Arbeiten mit einer Datenbank ein gültiges Benutzerkonto im ENI Server angelegt sein muss, und dass man sich darüber korrekt beim ENI-Server angemeldet haben muss. Standardmäßig werden die beim letzten Anmelden verwendeten Zugangsdaten gespeichert, bis erneut angemeldet wird.

Die Befehle:

- Quellcodeverwaltung ändern...
- Projekt aus Quellcodeverwaltung öffnen...
- Objekte aus der Quellcodeverwaltung hinzufügen
- Letze Version abrufen - Gesamtes Projekt
- Letzte Version abrufen - Ausgewählte Objekte
- Auschecken...
- Einchecken...
- Auschecken rückgängig
- Unterschiede anzeigen...
- Versionsgeschichte...
- Projektversionsgeschichte...
- Bezeichnung...
- Status aktualisieren
- Anmelden...

Beachten Sie auch den Befehl 'Anstehende Eincheckvorgänge' (Befehlskategorie 'Ansicht').

12.28.2 Quellcodeverwaltung ändern...

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') öffnet den Dialog **Quellcodeverwaltung ändern**. Hier kann die Verbindung zu einer externen Projektdatenbank über ENI aufgesetzt und aktiviert werden. Weiterhin können die Verknüpfungen der Projektobjekte zu den Ablageorten in der Datenbank angezeigt und verändert werden.

Abb. 256 Dialog 'Quellcodeverwaltung ändern'

Quellcodeverwaltung für dieses Projekt verwenden (ENI): Wenn diese Option aktiviert ist, wird das Projekt unter Quellcodeverwaltung gestellt. Das heißt, dass die Projektobjekte in deiner externen Datenbank verwaltet werden, mit der über einen ENI-Server kommuniziert wird. Die Verbindung zum ENI-Server wird im Feld 'Datenbankverbindung' definiert.:

'Datenbankverbindung':

Host: Adresse des Rechners, auf dem der ENI-Server läuft; z.B. "192.168.100.18", "testpc-1", "localhost".

Port: Diese Port-Nummer muss dieselbe sein, wie in den Konfigurationsparametern des ENI-Servers definiert: Default: 80.

Ort: Name des Datenbankordners, in dem die Projektobjekte per Default verwaltet werden sollen.

Wenn ein **bereits existierender Ordner**,verwendet werden soll, kann über Schaltfläche [...] der Dialog **Nach Speicherort suchen** geöffnet werden, der in Baumstruktur die bereits verfügbaren Ordner in der Datenbank anzeigt. Wählen Sie den gewünschten aus und bestätigen mit OK, um ihn im Feld 'Ort' einzutragen. Wenn der Dialog 'Quellcodeverwaltung ändern' mit OK geschlossen wird, erhalten Sie die Meldung, dass der gewählte Speicherort bereits verwendet wird, und dass das darin verwaltete Projekt ev. vom aktuellen abweicht. Zu diesem Zeitpunkt können Sie endgültig entscheiden, ob der Speicherort trotzdem verwendet werden soll.

Wenn ein **neuer Speicherort** für die Verwaltung des aktuellen Projekts definiert werden soll, geben Sie einen gewünschten Namen dafür ein. Wenn der Dialog 'Quellcodeverwaltung ändern' mit OK geschlossen wird, werden Sie nochmals gefragt, ob der Ordner mit diesem Namen in der Datenbank angelegt werden soll.

Nach Bestätigen des Speicherorts erscheint ein Dialog, in dem gefragt wird, ob alle Objekte des Projekts dorthin verknüpft werden sollen..

Wenn Ja, werden die Objekte im POUs- und Geräte-Fenster mit Symbolen versehen. Die dem Standard-Speicherort zugewiesenen Objekte erhalten gelbe Symbole: PLC_PRG (PRG) . Wenn ein Objekt bereits in der Datenbank gefunden wird, wird man jetzt gefragt, ob es dort durch das aktuelle aus dem Projekt überschrieben werden soll.

Wenn Nein, bleiben die Objekte "unverknüpft", d.h. sie werden nicht in der Datenbank abgelegt. Sie können aber auch später noch hinzugefügt werden (via Dialog Objekt Eigenschaften) bzw. der "Verknüpfungsstatus" kann in den 'Erweiterten Einstellungen' wie im folgenden beschrieben geändert werden:

Erweiterte Einstellungen: Diese Schaltfläche öffnet den Dialog 'Erweiterte Einstellungen zur Quellcodeverwaltung'. Hier findet man eine Übersicht über die **Verknüpfungen aller Projektobjekte**.

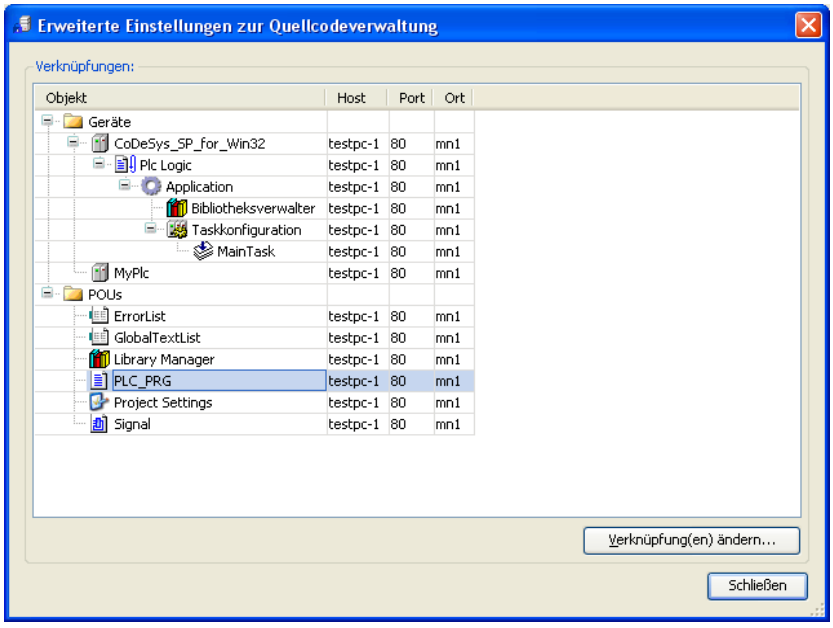


Abb. 257 Dialog 'Erweiterte Einstellungen zur Quellcodeverwaltung'

Für jedes Objekt sehen Sie die aktuell definierten Speicherort in der Projektdatenbank (Host, Port, Ort). Um eine oder mehrere Verknüpfungen zu bearbeiten, wählen Sie eine oder mehrere Objekte (Zeilen) aus und betätigen Schaltfläche **Verknüpfung(en) ändern...** um den Dialog 'Verknüpfung zu Quellcodeverwaltung ändern' zu öffnen. Eine Mehrfachselektion ist möglich über <Strg>+Mausklick oder durch Ziehen eines Rahmens mit der Maus um die gewünschten Zeilen. Wenn nur die Verknüpfung eines einzelnen Objekts geändert werden soll, genügt ein Doppelklick auf die entsprechende Zeile.

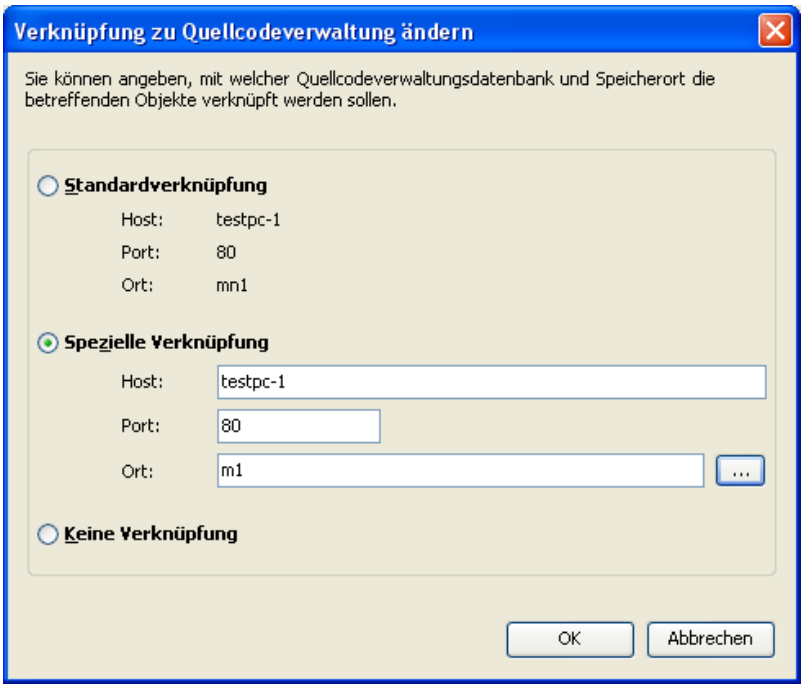


Abb. 258 Dialog 'Verknüpfung zu Quellcodeverwaltung ändern'

Im Dialog 'Verknüpfung zu Quellcodeverwaltung ändern' aktivieren Sie eine der folgenden Verknüpfungs-Optionen:

Standardverknüpfung: Wenn diese Option aktiviert wird, werden die ausgewählten Objekte mit dem Standard-Speicherort in der Datenbank verknüpft, der im oberen Teil des Dialogs (Datenbankverbindung) definiert ist. Hinweis: Die aktuelle Version der betroffenen Objekte wird außerdem auch am bisherigen Speicherort behalten, d.h. durch die Verknüpfungsänderung wird sie an den Standard-Speicherort **kopiert**!. Ein Objekt mit Standardverknüpfung ist im POUs- oder Geräte-Fenster durch ein gelbes Symbol gekennzeichnet, z.B.: PLC_PRG (PRG) .

Spezielle Verknüpfung: Wenn diese Option aktiviert wird, werden werden die ausgewählten Objekte mit einem anderen als dem Standard-Speicherort (Standardverknüpfung) verknüpft. Zu den Einstellungen Host, Port und Ort siehe oben bei 'Datenbankverbindung'. Diese Option kann beispielsweise verwendet werden, um die Funktionalität der Datenbank für "Gemeinsame Objekte" zu ersetzen, die in »PLC Designer« V2.3 verwendet wurde. Wenn bereits ein Objekt mit gleichem Namen im hier gewählten Speicherort gefunden wird, erscheint ein Dialog, in dem die möglichen Weiterwege aufgezeigt werden. Hinweis: Wenn ein Objekt hier "speziell verknüpft" wird, wird es am Standard-Speicherort gelöscht, d.h. es wird zum speziellen Speicherort **verschoben**! Ein Objekt mit spezieller Verknüpfung wird im POUs- oder Geräte-Fenster durch ein blaues Symbol gekennzeichnet, z.B.:

Keine Verknüpfung: Wenn diese Option aktiviert wird, sind die ausgewählten Objekte künftig nicht unter Quellcodeverwaltung. Das heißt, sie werden nur im Projekt gespeichert, was der n »PLC Designer« V2.3.verwendeten Objektkategorie "Lokal" entspricht.

12.28.3 Projekt aus Quellcodeverwaltung öffnen...

Dieser Befehl (Kategorie 'Quellcodeverwaltung') öffnet ein Projekt, das in einer ENI Projektdatenbank verwaltet wird. Der Dialog **Projekt aus Quellcodeverwaltung öffnen...** wird geöffnet, wo sie die entsprechenden Verbindungsparameter zur Datenbank und einen lokalen Speicherort für die Projektdatei angeben müssen.

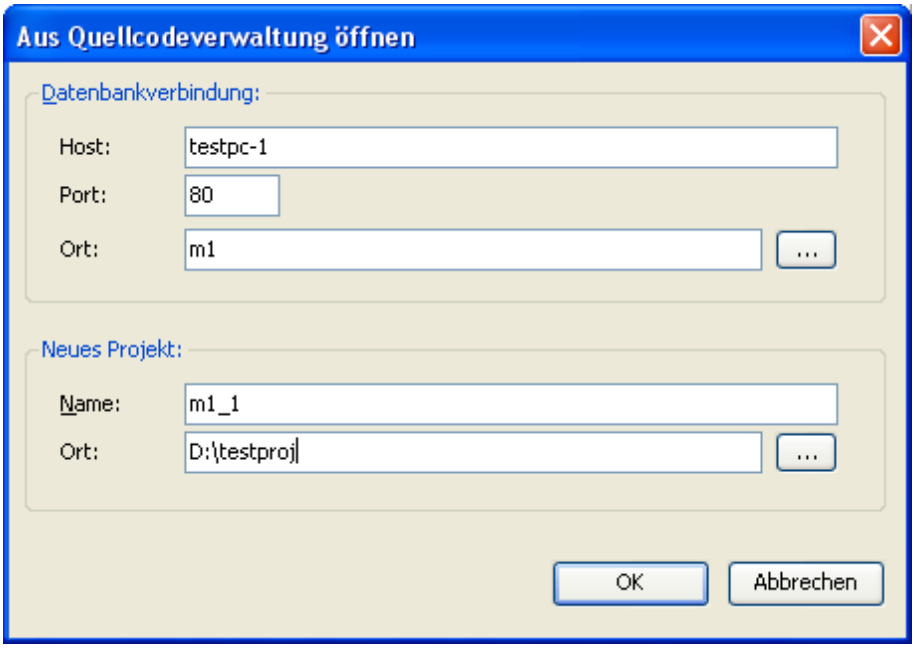


Abb. 259 'Projekt aus Quellcodeverwaltung öffnen...'

Datenbankverbindung:

Geben Sie hier die Verbindungsparameter für die Datenbank an, aus der Sie das Projekt öffnen wollen. Bezüglich der Einstellungen **Host**, **Port**, und **Ort** sehen Sie bitte in der Beschreibung zu Quellcodeverwaltung ändern.... Wenn der angegebene Speicherort kein Projekt enthält, erscheint eine entsprechende Meldung.

Neues Projekt:

Geben Sie einen **Name**n für das Projekt an, wie es auf Ihrem lokalen System heißen soll und definieren Sie den lokalen Ablage**Ort**. Dazu kann über Schaltfläche ... nach einem geeigneten Ordner gesucht werden.

Wenn ein gleichnamiges Projekt im definierten Ordner bereits vorliegt, erscheint eine entsprechende Meldung.

12.28.4 Objekte aus der Quellcodeverwaltung hinzufügen

Dieser Befehl (Kategorie 'Quellcodeverwaltung') dient dazu, dem gerade geöffneten Projekt einzelne Objekte aus der Datenbank hinzuzufügen. Dazu wird der Dialog **Objekte aus der Quellcodeverwaltung hinzufügen** geöffnet, wo in der Datenbank nach den gewünschten Objekten gesucht werden kann. Mehrfachselektion ist möglich.

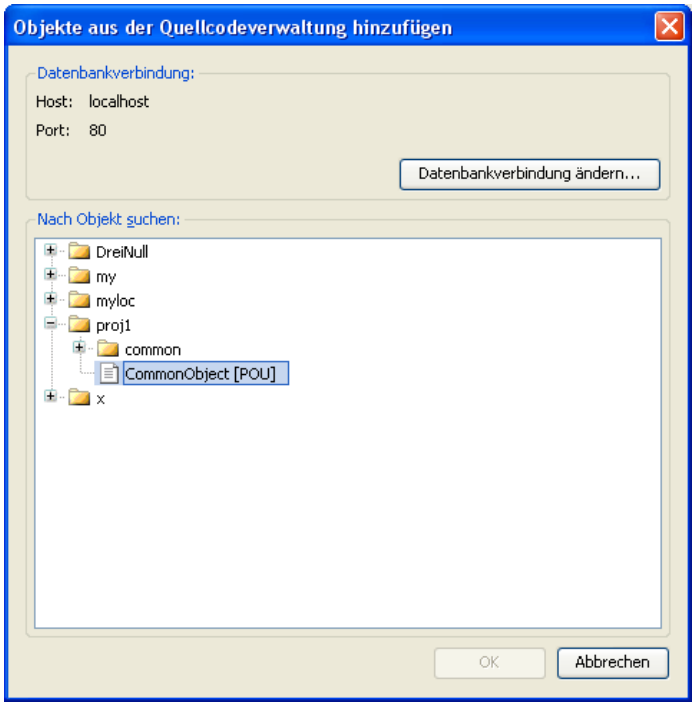


Abb. 260 Objekte aus der Quellcodeverwaltung hinzufügen'

Wenn Sie Objekte aus einer anderen ENI Datenbank als der gerade angebundenen benötigen, betätigen Sie die Schaltfläche **Datenbankverbindung ändern...** um im gleichnamigen Dialog den Namen (**Host**) und **Port** des entsprechenden ENI-Servers einzugeben. Nach erfolgreichem Anmelden auf diesem Server werden die Ordner der gewählten Datenbank angezeigt.

Die im Baum ausgewählten Objekte werden nach Bestätigen mit OK dem POUs- bzw. Gerätebaum des Projekts hinzugefügt. Die Standardverknüpfung ist automatisch die zu der Datenbank, aus der die Objekte geladen wurden und alle hinzugefügten Objekte (Ausnahme: Bibliotheksverwalter-Objekte) werden automatisch dorthin verknüpft. Um die Verknüpfungen zu ändern, verwenden Sie einen der Dialoge Objekt Eigenschaften bzw. Quellcodeverwaltung ändern....

12.28.5 Letze Version abrufen - Gesamtes Projekt

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') ruft die letzte Version aller Projektobjekte aus der Datenbank ab und überschreibt damit die lokale Version der Objekte im Projekt. Im Unterschied zum Auscheck-Vorgang werden die Objekte in der Datenbank nicht für andere Benutzer gesperrt.

12.28.6 Letzte Version abrufen - Ausgewählte Objekte

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') ruft die letzte Version aller Projektobjekte, die gerade im POUs- oder Geräte-Fenster ausgewählt sind, aus der Datenbank ab und überschreibt damit die lokale Version. Im Unterschied zum Auscheck-Vorgang werden die Objekte in der Datenbank nicht für andere Benutzer gesperrt.

12.28.7 Auschecken...

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') checkt die gerade im POUs- oder Geräte-Fenster ausgewählten Objekte aus der Datenbank aus und sperrt sie damit in der Datenbank für andere Benutzer.

Der Dialog 'Auschecken' wird geöffnet, wo ein **Kommentar** zum Vorgang eingegeben werden kann, der in der Versionshistorie des Objekts gespeichert wird. Zeilenumbrüche können dabei mit <Strg>+<Eingabe> eingefügt werden.

Wenn die Option **Keine lokale Kopie abrufen** aktiviert wird, wird das Objekt in der Datenbank als "ausgecheckt" markiert, aber die letzte Version wird nicht automatisch ins Projekt kopiert. Dies kann nützlich sein, wenn das Objekt zwar für andere Benutzer gesperrt, die lokale Version im Projekt aber nicht durch die Datenbankversion überschrieben werden soll.

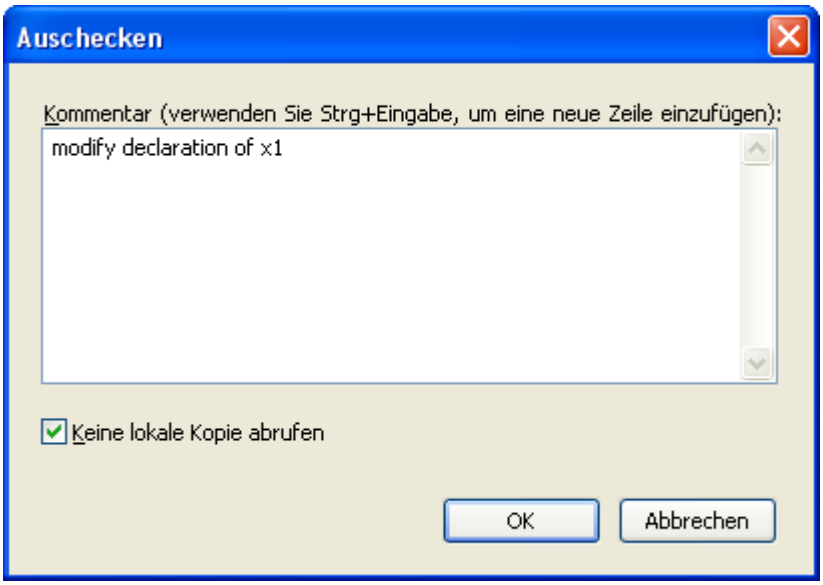


Abb. 261 Dialog 'Auschecken'

Wenn sich die Datenbankversion des Objekts von der im Projekt vorliegenden unterscheidet, erscheint eine entsprechende Meldung und der Benutzer kann nochmals entscheiden, ob das Objekt ausgecheckt werden soll.

Nachdem der Dialog mit OK geschlossen wurde, wird das ausgecheckte Objekt mit einem **grünen Häkchen** im POUs- oder Gerätebaum des Projekts gekennzeichnet. Für andere Benutzer, die das Projekt bearbeiten, erscheint es mit einem **roten Kreuz** versehen und kann nicht bearbeitet werden. Für das Einchecken eines Objekts siehe Befehl Checkin...

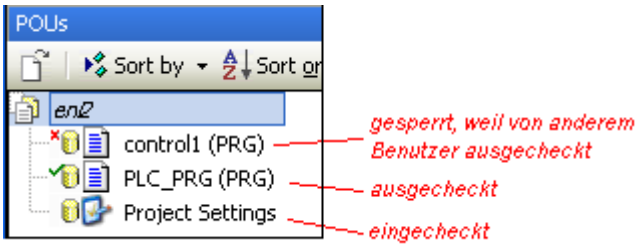


Abb. 262 Symbole für ausgecheckte Objekte

12.28.8 Einchecken...

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') checkt die ausgecheckten Objekte, die gerade im POUs- oder Geräte-Fenster ausgewählt sind, in die Projektdatenbank ein. Dadurch wird in der Datenbank eine neue Version des Objekts angelegt. Die alten Versionen werden jedoch ebenfalls weiterhin gespeichert.

Der Dialog 'Einchecken' wird geöffnet, wo ein **Kommentar** für den Vorgang eingegeben werden kann, der in der Versionshistorie des Objekts gespeichert wird. Zeilenumbrüche können dabei mit <Strg>+<Eingabe> eingefügt werden.

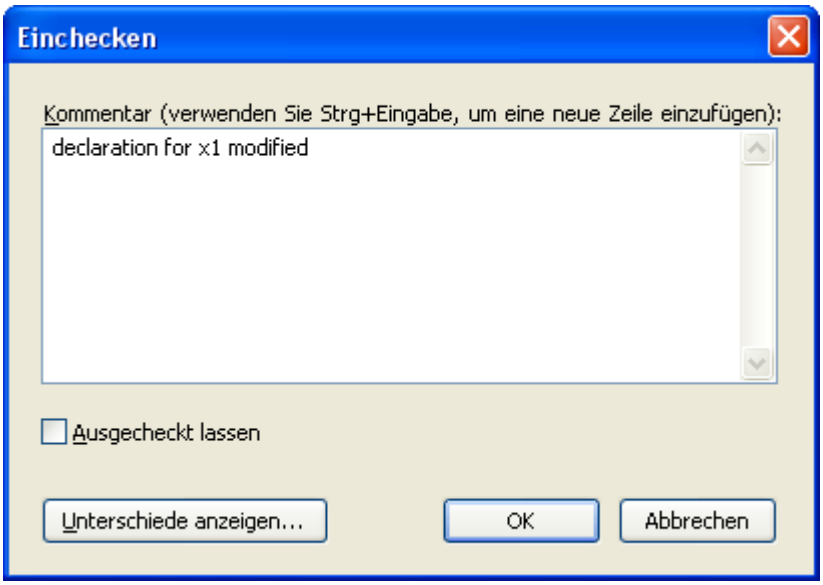


Abb. 263 Dialog 'Einchecken'

Ausgecheckt lassen: Wenn diese Option aktiviert ist, wird die aktuelle Version des Objekts in die Datenbank geschrieben, aber das Objekt bleibt weiterhin im Status "ausgecheckt".

Unterschiede anzeigen...: Wenn diese Schaltfläche betätigt wird, erscheint eine Meldungsbox mit der Information, ob sich die lokale Version des Objekts von der letzten in der Datenbank unterscheidet. (Derzeit werden die Unterschiede noch nicht im Detail angezeigt, sondern nur eine Information, ob welche bestehen.)

Nachdem der Dialog mit OK geschlossen wurde und Option 'Ausgecheckt lassen' nicht aktiviert ist, wird das grüne Häkchen vor dem Objektnamen im POUs- oder Gerätefenster entfernt.

Beachten Sie auch den Dialog 'Anstehende Eincheckvoränge', der zur Anzeige und Handhabung der aktuell ausgecheckten Objekte verwendet werden kann.

12.28.9 Auschecken rückgängig

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') macht das Auschecken für die gerade im POUs- oder Geräte-Fenster ausgewählten Objekte rückgängig.

Wenn in der lokalen Objektversion Änderungen vorgenommen wurden, werden Sie gefragt, ob die lokale Version jetzt durch die letzte Datenbankversion überschrieben werden soll, d.h. ob die vorgenommenen Änderungen verworfen werden sollen. Die Objektversion in der Datenbank bleibt unverändert und das Objekt wird für andere Benutzer wieder editierbar. Das rote Kreuz vor dem Objektnamen im POUs- oder Geräte-Fenster verschwindet.

12.28.10 Unterschiede anzeigen...

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') gibt Information darüber, ob sich die lokale Version des Objekts von der letzten Version des Objekts in der Datenbank unterscheidet.

(Derzeit werden die Unterschiede noch nicht im Detail angezeigt, sondern nur eine Information, ob welche bestehen.)

12.28.11 Versionsgeschichte...

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') zeigt eine Tabelle mit den Datenbankaktionen (z.B. Einchecken, Auschecken, Bezeichnung etc.), die bis jetzt für das gerade im POUs- oder Geräte-Fenster ausgewählte Objekt durchgeführt wurden. (Siehe auch 'Projektversionsgeschichte' für das gesamte Projekt.)

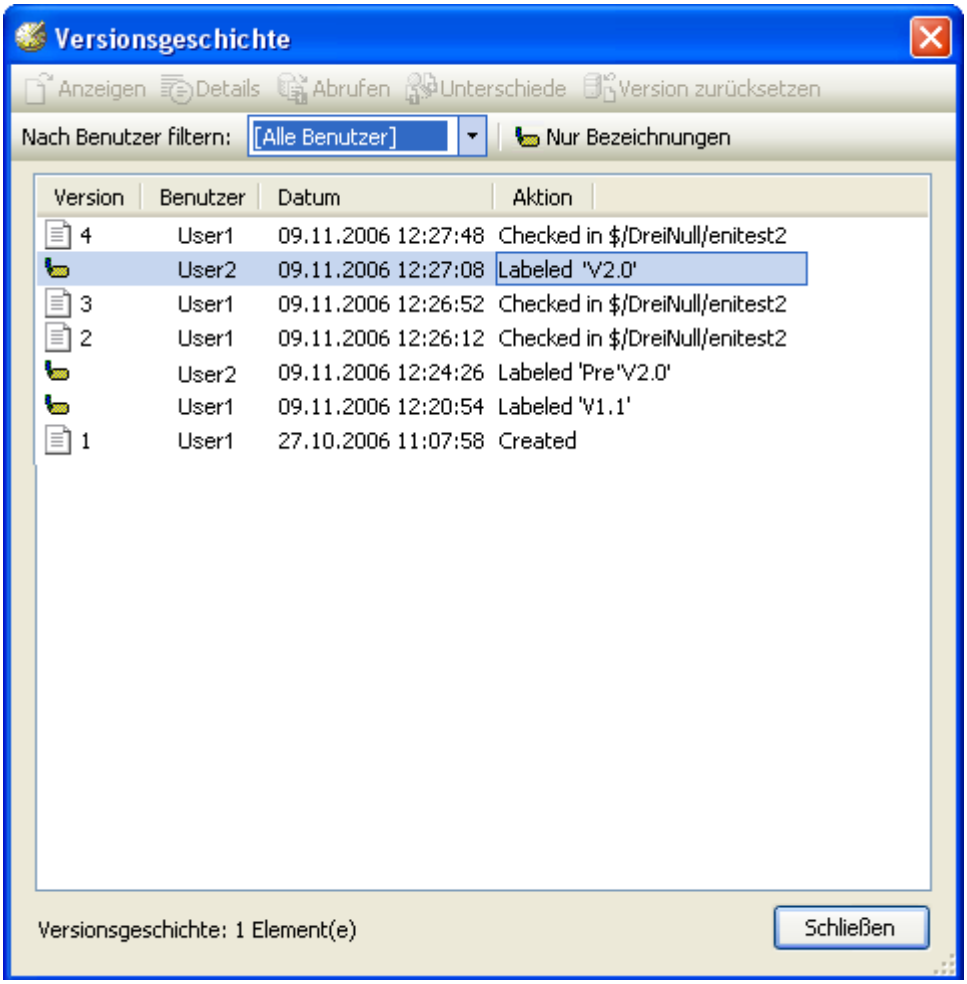


Abb. 264 Dialog 'Versionsgeschichte', Beispiel

Die Tabelle enthält in chronologischer Reihenfolge von unten nach oben für jede Datenbankaktion eine Zeile mit folgenden Informationen:

Version: Datenbankspezifische Nummerierung der Versionen des Objekts, die nacheinander eingecheckt wurde. Bezeichnete (Label) Versionen erhalten keine Versionsnummer, sondern sind mit einem Bezeichner-Symbol ( ) gekennzeichnet.

Benutzer: Name des Benutzers, der die Eincheck- oder Bezeichner-Aktion vorgenommen hat.

Datum: Datum und Uhrzeit der Aktion.

Aktion: Typ der Aktion, die am Objekt durchgeführt wurde. Die Angabe ist datenbankabhängig, z.B. 'erstellt' (das Objekt wurde in der Datenbank erstmals eingecheckt), 'eingecheckt' oder 'bezeichnet mit <Bezeichner>' (diese Version des Objekts wurde mit einem Bezeichner versehen).

Durch einen Doppelklick auf einen Eintrag in der Liste wird der Dialog **Versionsgeschichtedetails** geöffnet. Siehe auch unten, Befehl "Details".

Mögliche Filter für die Tabelle der Einträge (in der Leiste direkt über der Tabelle):

Nach Benutzer filtern: Standardmäßig werden die Aktionen aller Benutzer hier aufgelistet, es kann jedoch ein Filter für einen bestimmten Benutzernamen gesetzt werden.

Nur Bezeichnungen: Mit dieser Schaltfläche kann zwischen der Anzeige ausschließlich der Bezeichnervergabe-Aktionen und der Anzeige allen Aktionen hin- und hergeschaltet werden.

Befehle (als Schaltflächen in der Leiste ganz oben im Dialog):

Anzeigen:	Die in der Tabelle markierte Version wird im entsprechenden Editor angezeigt, kann aber nicht bearbeitet werden.
Details:	Der Dialog 'Details der Versionsgeschichte' öffnet und gibt folgende Informationen: **Objekt**name, **Version** (s.o.), **Bezeichnung** (wenn vergeben, s.o.) **Datum** (s.o.), **Benutzer** (s.o.), **Aktion** (s.o.), **Kommentar** (Kommentar, der beim Einchecken bzw. bei der Bezeichnervergabe eingegeben wurde). Über die Schaltflächen **Nächste >** bzw. **< Vorherige** kann zu den Details des nächsten bzw. vorherigen Eintrags im Dialog 'Versionsgeschichte' gesprungen werden.
Abrufen:	Die in der Tabelle markierte Version wird aus der Datenbank ins Projekt geladen und überschreibt die lokale Version.
Unterschiede:	Zwei Einträge in der Liste können ausgewählt werden und mit Hilfe dieses Befehls verglichen werden. (Derzeit werden die Unterschiede noch nicht im Detail angezeigt, sondern nur eine Information, ob welche bestehen.)
Version zurücksetzen:	Die in der Tabelle markierte Version wird als aktuelle, also letzte Datenbankversion gesetzt. Die später eingefügten Versionen werden gelöscht ! Dies kann benützt werden, um einen früheren Stand wiederherzustellen und als aktuellen zu führen.

12.28.12 Projektversionsgeschichte...

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') zeigt eine Tabelle mit den Datenbankaktionen (z.B. Einchecken, Auschecken, Bezeichnung etc.), die bis jetzt für das gesamte Projekt durchgeführt wurden. Sie werden gleich gefragt, ob die einzelnen Objekt-Versionsgeschichten mit angezeigt werden sollen, diese können jedoch auch später noch mit Hilfe des Befehls 'Objekte anzeigen' ein- oder ausgeblendet werden, s.u.

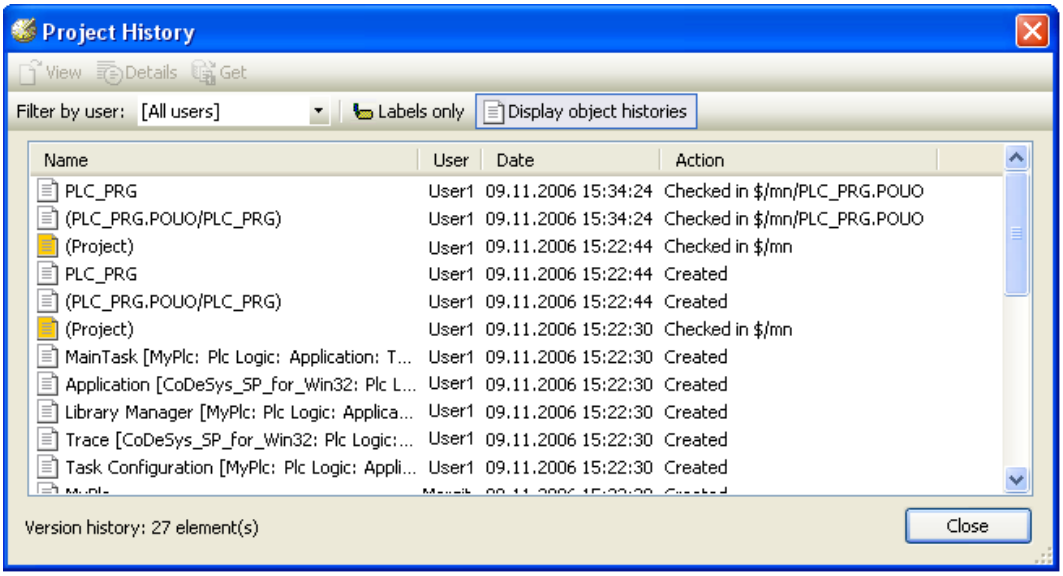


Abb. 265 Dialog 'Projektversionsgeschichte', Beispiel

Die Tabelle enthält in chronologischer Reihenfolge von unten nach oben für jede Datenbankaktion eine Zeile, die im Unterschied zur Objekt-Versionsgeschichte nicht numeriert ist. Für jede Aktion ist folgende Information gegeben:

Name: Name des Projekts bzw. Objekts.

Benutzer: Name des Benutzers, der die Eincheck- oder Bezeichner-Aktion vorgenommen hat.

Datum: Datum und Uhrzeit der Aktion.

Aktion: Typ der Aktion, die am Objekt durchgeführt wurde. Die Angabe ist datenbankabhängig, z.B. 'erstellt' (das Objekt wurde in der Datenbank erstmals eingecheckt), 'eingecheckt' oder 'bezeichnet mit <Bezeichner>' (diese Version des Objekts wurde mit einem Bezeichner versehen)

Die Anzahl der im Projekt enthaltenen Datenbankobjekte wird unterhalb der Liste angezeigt.

Durch einen Doppelklick auf einen Eintrag in der Liste wird der Dialog **Versionsgeschichtedetails** geöffnet. Siehe auch unten, Befehl "Details".

Mögliche Filter für die Tabelle der Einträge (in der Leiste direkt über der Tabelle):

Nach Benutzer filtern: Standardmäßig werden die Aktionen aller Benutzer hier aufgelistet, es kann jedoch ein Filter für einen bestimmten Benutzernamen gesetzt werden.

Nur Bezeichnungen: Mit dieser Schaltfläche kann zwischen der Anzeige ausschließlich der Bezeichnervergabe-Aktionen und der Anzeige allen Aktionen hin- und hergeschaltet werden.

Objekte anzeigen: Mit dieser Schaltfläche kann zwischen der Anzeige ausschließlich der Aktionen, die das gesamte Projekt betreffen und der Anzeige dieser Aktionen plus der Versionsgeschichten der einzelnen Objekte hin- und hergeschaltet werden. Im letzteren Fall werden die einzelnen objektbezogenen Aktionen in chronologischer Reihenfolge zwischen den projektbezogenen eingefügt. Die Objektaktionen sind mit einem -Symbol versehen, die Projektaktionen mit einem -Symbol.

Befehle (als Schaltflächen in der Leiste ganz oben im Dialog):

Anzeigen:	Die in der Tabelle markierte Version wird im entsprechenden Editor angezeigt, kann aber nicht bearbeitet werden.
Details:	Der Dialog 'Details der Versionsgeschichte' öffnet und gibt folgende Informationen: **Objekt**name, **Version** (s.o.), **Bezeichnung** (wenn vergeben, s.o.) **Datum** (s.o.), **Benutzer** (s.o.), **Aktion** (s.o.), **Kommentar** (Kommentar, der beim Einchecken bzw. bei der Bezeichnervergabe eingegeben wurde). Über die Schaltflächen **Nächste >** bzw. **< Vorherige** kann zu den Details des nächsten bzw. vorherigen Eintrags im Dialog 'Versionsgeschichte' gesprungen werden.
Abrufen:	Alle Objekte der gerade in der Tabelle ausgewählten markierte Version (Datum) werden aus der Datenbank in das Projekt geladen und **überschreiben** die lokale Version. **Zu beachten:** Dies heißt auch, dass lokale Objekte, die nicht in der abgerufenen Datenbank-Version aus dem Projekt gelöscht werden! Objekte, die in der abgerufenen Datenbank-Version enthalten sind, aber nicht im Projekt, werden dem Projekt hinzugefügt. Wenn die abgerufene Version andere Verknüpfungen für Datenbankobjekte definiert als die im Projekt, werden die Definitionen im Projekt durch die aus der Datenbank überschrieben.

12.28.13 Bezeichnung...

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') öffnet den Dialog 'Bezeichnung', in dem für die aktuelle Version der Datenbankobjekte im lokalen Projekt eine Bezeichnung ("Label") vergeben werden kann.

Durch Angabe dieser Bezeichnung kann später genau diese Version der Objekte wieder aus der Datenbank abgerufen werden.

Geben Sie im Dialog eine **Bezeichnung** ein (z.B.. "Release Version 1.0", "Testversion xy") und optional einen **Kommentar**. Nach Bestätigen mit OK schließt der Dialog und die Bezeichnervergabe-Aktion wird in der Projektversionsgeschichte wie auch in der Versionsgeschichte jedes betroffenen Objekts hinzugefügt. Eine bezeichnete Version erhält keine Versionsnummer, sondern nur ein Bezeichner-Symbol in der Spalte 'Version'. Wenn die Option 'Nur Bezeichnungen' in einem Versionsgeschichte-Dialog aktiviert ist, werden nur die Bezeichnervergabeaktionen, also die bezeichneten Versionen, aufgelistet.

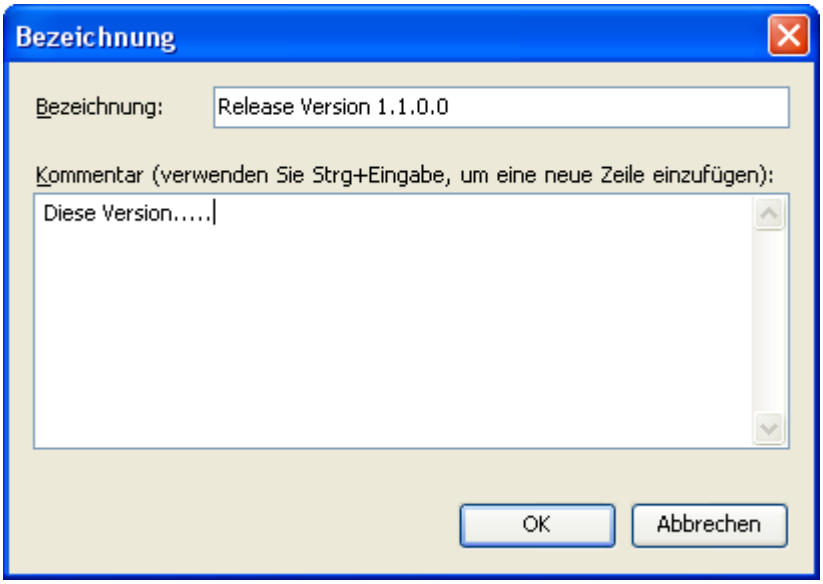


Abb. 266 Dialog 'Bezeichnen', Beispiel

12.28.14 Status aktualisieren

Symbol:

Dieser Befehl (Kategorie 'Quellcodeverwaltung') veranlaßt eine Prüfung des aktuellen Quellcodeverwaltungsstatus aller Objekte. Die entsprechenden Symbole werden im POUs- und Geräte-Fenster aktualisiert und Informationen und ggfs. Fehlermeldungen oder Warnungen zur Status-Aktualisierung werden im Meldungsfenster ausgegeben.

12.28.15 Anmelden...

Dieser Befehl (Kategorie 'Quellcodeverwaltung') öffnet den Dialog **Bei Quellcodeverwaltung anmelden**, wo man sich über den ENI Server in die Projektdatenbank einloggen kann. Voraussetzung dafür ist ein gültiges Benutzerkonto im ENI Server.

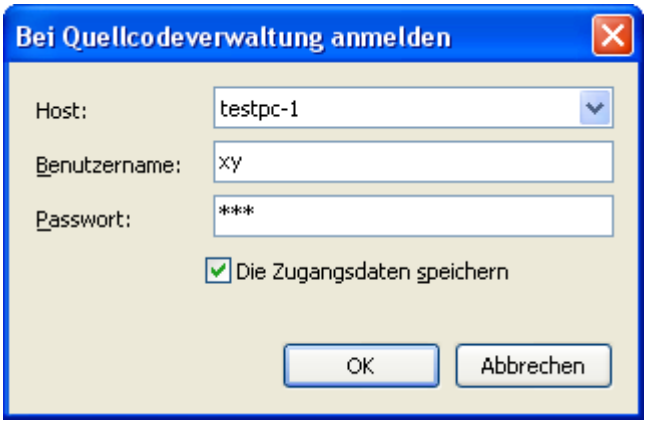


Abb. 267 Dialog 'Bei Quellcodeverwaltung anmelden'

Host: Hier muss der Name bzw. die TCP/IP-Adresse des ENI-Servers eingegeben werden, der die Verbindung zur externen Projektdatenbank herstellt.

Benutzername, **Passwort**: Zugangsdaten für Ihr Benutzerkonto im ENI-Server.

Die Zugangsdaten speichern: Wenn diese Option aktiviert ist, werden die oben angegebenen Zugangsdaten gespeichert, bis der Anmelden-Dialog erneut verwendet wird. Wenn die Option nicht aktiviert ist, werden Benutzername und Passwort nur bis zum Beenden von »PLC Designer« gespeichert. Beachten Sie, dass im Unterschied zu »PLC Designer« V2.3 die Anmeldung nur einmal, nämlich beim betreffenden ENI-Server, durchzuführen ist, und nicht separat für jedes einzelne Projekt bzw. jede einzelne Objektkategorie.

12.29 Text Editor

Die Befehlskategorie 'TextEditor' enthält Befehle zum Arbeiten in den Text-Editoren.

12.29.1 Befehle der Kategorie Texteditor

Das Plug-In 'Text Editor Basics' stellt Befehle der Kategorie 'Texteditor' für das Bearbeiten eines Texts im Projekt bereit. Standardmäßig sind diese Befehle im Untermenü "Advanced" im Bearbeiten-Menü zu finden. Wenn erforderlich, kann die Menükonfiguration über den Anpassen-Dialog verändert werden.

Die Befehle:

- Überschreibmodus
- Gehe zu Zeile...
- Datei als Text einfügen...
- In Kleinbuchstaben umwandeln
- In Großbuchstaben umwandeln
- Gehe zur zugehörigen Klammer
- Bis zur zugehörigen Klammer selektieren

12.29.2 Überschreibmodus

Diese Option ('Kategorie 'Texteditor') kann aus- und ein-, d.h. zwischen Überschreibmodus und Einfügemodus hin- und hergeschaltet werden. Wenn im Überschreibmodus gearbeitet wird, werden die vorhandenen Zeichen durch die neu eingegebenen überschrieben, ansonsten werden die neuen Zeichen zwischen den bestehenden eingefügt.

12.29.3 Gehe zu Zeile...

Mit diesem Befehl ('Kategorie 'Texteditor') kann zu einer bestimmten Zeile innerhalb des aktiven Texteditors gesprungen werden. Dazu wird der Dialog 'Gehe zu Zeile' geöffnet, in dem man die gewünschte Zeilennummer eingibt. Nach Schließen dieses Dialogs mit OK wird der Mauszeiger an den Beginn der entsprechenden Zeile gesetzt.

12.29.4 Datei als Text einfügen...

Mit diesem Befehl ('Kategorie 'Texteditor') kann der Inhalt einer Textdatei in den gerade aktiven Texteditor kopiert werden. Der Standarddialog zum Öffnen einer Datei ('Datei einfügen') wird geöffnet. Suchen Sie die gewünschte Datei, die im Textformat verfügbar sein muss. Der Dateiinhalt wird dann an der aktuellen Cursor-Position eingefügt.

12.29.5 In Großbuchstaben umwandeln

Mit diesem Befehl ('Kategorie 'Texteditor') wird der gerade selektierte Textbereich in Großbuchstaben dargestellt.

12.29.6 In Kleinbuchstaben umwandeln

Mit diesem Befehl ('Kategorie 'Texteditor') wird der gerade selektierte Textbereich in Kleinbuchstaben dargestellt.

12.29.7 Gehe zur zugehörigen Klammer

Mit diesem Befehl ('Kategorie 'Texteditor') kann, wenn der Cursor bei einer Klammer steht, zur zugehörigen öffnenden bzw. schließenden Klammer gesprungen werden. Dies gilt für Klammern innerhalb von Programmzeilen wie auch "Klammerungen", die über mehrere Zeilen reichen. Die Anzeige der Klammerungen kann mit der gleichnamigen Option im Optionen-Dialog, Kategorie Texteditor/Randbereich aktiviert werden.

12.29.8 Bis zur zugehörigen Klammer selektieren

Mit diesem Befehl ('Kategorie 'Texteditor') werden, wenn der Cursor bei einer Klammer steht, alle Code-Zeilen bis zur entsprechenden schließenden oder öffnenden Klammer selektiert. Die Selektion kann dabei also aufwärts oder abwärts erfolgen. Dies gilt für Klammern innerhalb von Programmzeilen wie auch "Klammerungen", die über mehrere Zeilen reichen. Die Anzeige der Klammerungen kann mit der gleichnamigen Option im Optionen-Dialog, Kategorie Texteditor/Randbereich aktiviert werden.

12.30 Textlisten

Die Befehlskategorie 'Textlisten' enthält Befehle zum Erstellen, Bearbeiten, Exportieren und Imporitieren von Textlisten.

12.30.1 Befehle der Kategorie Textlisten

Die Befehle zum Arbeiten mit Textlisten werden mit dem Plug-In 'Text lists for multi language support' bereitgestellt.

Standardmäßig sind diese Befehle in einem Menü 'Textliste' verfügbar, sobald eine Textliste bearbeitet wird, bzw. zum Teil im Menü 'Visualisierung'. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs verändert werden.

Die Befehle:

- Globale Textliste anlegen
- Sprache einfügen
- Sprache entfernen
- Import/Export Textlisten
- Visualisierungs-Text-IDs aktualisieren
- Visualisierungs-Text-IDs prüfen
- Nicht verwendete Textlisteneinträge entfernen

12.30.2 Globale Textliste anlegen

Symbol:

Dieser Befehl der Kategorie Textliste dient dazu, **explizit** eine globale Textliste anzulegen. Er steht standardmäßig im Menü '**Visualisierung**' bereit. Das Objekt GlobalTextList wird im POUs-Fenster eingefügt.

Automatisch wird GlobalTextList angelegt, sobald beim Konfigurieren einer Visualisierung der erste Text definiert wird. Siehe auf Hilfeseite Textliste detaillierte Informationen zu Textlisten.

12.30.3 Sprache einfügen

Symbol:

Dieser Befehl der Kategorie Textliste dient dazu, in einer Textliste eine Spalte für eine zusätzliche Landessprache einzufügen.

Dazu muss die Textliste im Editor geöffnet und der Befehl ausgeführt werden. Der Dialog **Sprache eingeben** wird geöffnet, wo ein Name (Spaltenüberschrift in Textliste) für die neue Sprache einzugeben und mit OK zu bestätigen ist. Daraufhin wird die entsprechende Spalte in der Textliste angelegt.

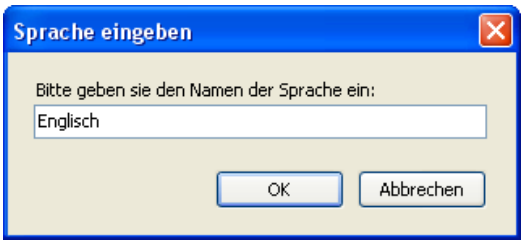


Abb. 268 Dialog 'Sprache eingeben'

12.30.4 Sprache entfernen

Symbol:

Dieser Befehl der Kategorie Textliste dient dazu, aus einer Textliste die Spalte für eine bestimmte Sprache zu entfernen. Dazu muss der Cursor in eine der Spaltenzellen gesetzt und der Befehl ausgeführt werden.

12.30.5 Import/Export Textlisten

Symbol:

Dieser Befehl der Kategorie Textliste dient dem Datenaustausch mit anderen Programmen wie z.B. Excel. Als Datenformat wird .csv ("Comma Separated Values") verwendet. Bei Aufruf des Kommandos erscheint folgender Dialog:

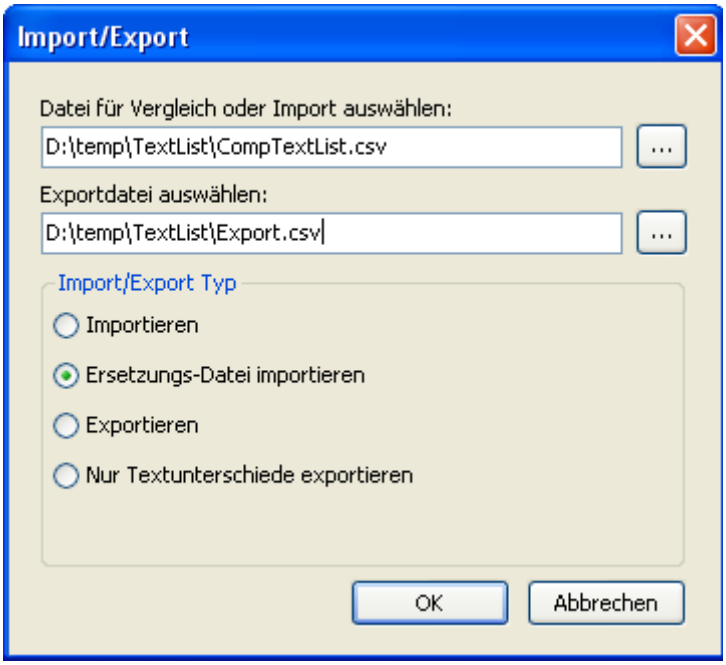


Abb. 269 Dialog - Import/Export Textlisten

Durch Angabe des entsprechenden Pfads oder durch Zuhilfenahme der Eingabehilfe (...) können die Dateien für den Import, Export oder für einen Vergleich bestimmt werden. Welche der Aktionen durchgeführt werden soll, kann durch eine entsprechende Aktivierung im unteren Teil des Dialogs bestimmt werden:

- **Importieren**: Beim Import einer externen Datei werden die Datensätze der externen Datei mit den Datensätzen des Projekts abgeglichen. Die Datensätze im Projekt werden nach folgenden Regeln angepasst:

1. Sind die Textinhalte der Dateien identisch, so gibt es keine Änderung am Datensatz.
2. Ist eine Übersetzung innerhalb der externen Datei hinzugekommen, so wird sie in den Datensatz im Projekt übernommen.
3. Erfolgte eine Textänderung innerhalb einer Übersetzung, so wird die Änderung in den Datensatz im Projekt übernommen.
4. Fehlen Übersetzungstexte innerhalb der externen Datei, so erfolgt keine Änderung am Datensatz im Projekt.
5. Ist eine neue Zeile innerhalb der externen Datei hinzugekommen, so wird der neue Datensatz in das Projekt aufgenommen.
6. Ist eine zusätzlicher Datensatz im Projekt enthalten, so bleibt der Datensatz erhalten.

7. Eine Änderung innerhalb der Spalte „Default“ kann wie das Einfügen eines neuen Textes betrachtet werden. Wenn Textstellen jedoch statt eines mehrere Leerzeichen aufweisen, wird dies nicht als Änderung betrachtet!

Beispiel - Import einer .cvs-Datei:

Dateninhalt der externen Datei:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	2	Light		

Dateninhalt der Textlisten des aktuellen Projekts vor dem Import:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abortion
TextList1	1	Door	Tür	Door
TextList2	3	Seat	Sitz	Seat

Beim Import werden alle Unterschiede in das Projekt aufgenommen. Dabei erfolgt ein Abgleich der beiden Listen, so dass der Inhalt der nachfolgenden Textlisten im Projekt entsteht:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door	Tür	Door
TextList1	2	Light		
TextList2	3	Seat	Sitz	Seat

- **Ersetzungsdatei importieren**: Beim Import von Textlisten wird eine Änderung innerhalb der Spalte „Default“ wie das Einfügen einer neuen Zeile betrachtet. Der Grund dafür ist, dass die Spalte „Default“ beim Export/Import als Schlüssel für den Vergleich von Zeilen dient. Hinweis: Wenn Textstellen in Spalte "Default" jedoch statt eines mehrere Leerzeichen aufweisen, wird dies nicht als Änderung betrachtet!
  Soll nun innerhalb der Spalte „Default“ der Text verändert werden (Beseitigung eines Schreibfehlers oder Ergänzung eines bestehenden Textes) so bedarf es einer Änderungsliste (Ersetzungsdatei).

Beispiel - Import einer Änderungsliste

Default alt	Default neu	Kommando
Cancel ?	Cancel	REPLACE
Möchten Sie Speichern ?	Möchten Sie wirklich Speichern ?	REPLACE_AND_REMOVE
Möchten Sie wirklich Speichern ?	Möchten Sie wirklich Speichern ?	REPLACE_AND_REMOVE

Die Änderungsliste wird von oben nach unten abgearbeitet. Dadurch kann auch eine Änderungshistorie berücksichtigt werden.

Das Kommando beschreibt was mit der Textzeile zu tun ist. Momentan steht nur das Kommando REPLACE stehen zur Verfügung, welches den folgenden Effekt hat:

Im Normalfall wird der in der Spalte "Default" eingetragene Text durch den neuen Text ersetzt. Im Beispiel wird „Cancel ?“ durch „Cancel“ ersetzt und "Möchten Sie Speichern ?" durch "Möchten Sie wirklich Speichern ?". Simultan werden die Texte aller Visualisierungselemente angepasst, d.h. auch hier werden die alten Einträge durch die neuen Texte ersetzt.

Im Fall, dass der neue Default Text schon in einem Feld der Spalte "Default" einer anderen Zeile der Textliste enthalten ist, wird die Zeile, die den zu ersetzenden Eintrag enthält, komplett gelöscht. Die betroffenen Visualisierungselemente erhalten die Einträge aus der verbleibenden Spalte mit demselben Default Eintrag. Im Beispiel tritt dieser Fall für den Eintrag „Möchten Sie wirklich Speichern ?!“, der durch „Möchten Sie wirklich Speichern ?“ ersetzt werden soll. Aufgrund der Änderungshistorie gibt es zu dem Zeitpunkt, da das Kommando REPLACE für den Eintrag „Möchten Sie wirklich Speichern ?!“ aufgerufen wird, aber schon einen Default Eintrag „Möchten Sie wirklich Speichern ?“. Um ein mehrfaches Vorkommen dieses Schlüssels zu vermeiden, wird die Zeile mit dem alten Default Eintrag „Möchten Sie wirklich Speichern ?!“ komplett aus der Textliste gelöscht.

- **Exportieren**: Beim Export von Textlisten werden alle Änderungen innerhalb des Projekts mit einer externen Vergleichs-Datei abgeglichen. Es wird eine neue Export-Datei erstellt, welche nachfolgende Regeln befolgt:

1. Die Textinhalte sind identisch, so wird der Datensatz unverändert exportiert.
2. Ist eine Übersetzung im Projekt hinzugekommen, so wird sie neu aufgenommen.
3. Erfolgte eine Textänderung innerhalb einer Übersetzung, so wird Textänderung aus dem Projekt für den neuen Datensatz verwendet.
4. Fehlen Übersetzungstexte im Projekt, so werden die Übersetzungen der Vorlage für den neuen Datensatz verwendet.
5. Ist eine neue Zeile im Projekt hinzugekommen, so wird sie als neuer Datensatz aufgenommen.
6. Ist in der externen Datei eine zusätzliche Zeile enthalten, so wird diese wieder exportiert.
7. Eine Änderung innerhalb der Spalte „Default“ kann wie das Einfügen eines neuen Textes betrachtet werden.

Beispiel - Export einer .cvs-Datei

Dateninhalt der externen Datei:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abort
TextList1	1	Door	Tür	Door
TextList2	2	Seat	Sitz	Seat

Dateninhalt der Textlisten des aktuellen Projekts vor dem Export:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering whee
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	3	Light		
TextList2				

Beim Export werden alle Unterschiede in die externe Datei aufgenommen. Dabei erfolgt ein Abgleich der beiden Listen sodass die nachfolgende externe Datei entsteht.

Dateninhalt der externen Datei nach dem Export:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door	Tür	Door
TextList1	3	Light		
TextList2	2	Seat	Sitz	Seat

- **Nur Textunterschiede exportieren**: Ist diese Option aktiviert, so werden nur die in den unterschiedlichen Versionen voneinander abweichenden Zeilen in die Exportdatei aufgenommen. Eine solche Differenzdateien eignet sich beispielsweise als Vorlage für neu zu erstellende Übersetzungen. Da sie möglichst klein gehalten werden soll, werden fehlende Einträge in den aktuellen Listen nicht als Abweichung gerechnet.

Beispiel - Export der Textunterschiede

Dateninhalt der externen Datei:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abort
TextList1	1	Door	Tür	Door
TextList2	2	Seat	Sitz	Seat

Dateninhalt der Textlisten des aktuellen Projekts vor dem Export:

TextList	Id	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	3	Light		
TextList2				

Beim Export werden alle Zeilen, die in den verschiedenen Versionen voneinander abweichen (Zeilen 2, 3 und 5 der aktuellen Liste), in die Exportdatei aufgenommen.

Dateninhalt der externen Datei nach dem Export:

TextList	Id	Default	Deutsch	English
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	3	Light		

Hinweis: Für das Auffinden des korrespondierenden Datensatzes wird für die GlobalTextList die Spalte „Default“ und für alle anderen Textlisten die Spalte „Id“ verwendet. Deshalb ist die Spalte „Id“ für alle Datensätze der GlobalTextList leer.

12.30.6 Visualisierungs-Text-IDs aktualisieren

Symbol:

Dieser Befehl gehört zur Kategorie Textliste. Wird ein statischer Text innerhalb eines Visualisierungselements verändert, so muss die Visualisierung und evtl. auch die GlobalTextList das Schreibrecht besitzen (sh. GlobalTextList). Wenn trotz fehlenden Schreibrechts Veränderungen vorgenommen werden, so kann es vorkommen, dass die Text-Ids nicht mehr zu den Texten innerhalb eines Visualisierungselements passen.

Mit Hilfe des Kommandos „Visualisierungs-Text-Ids aktualisieren“ können diese Fehlerfälle automatisch behoben werden. Dazu müssen alle Visualisierungen mit Fehlerfällen und die GlobalTextList das Schreibrecht besitzen.

12.30.7 Visualisierungs-Text-IDs prüfen

Symbol:

Dieser Befehl gehört zur Kategorie Textliste. Wird ein statischer Text innerhalb eines Visualisierungselements verändert, so muss die Visualisierung und evtl. auch die GlobalTextList das Schreibrecht besitzen (sh. GlobalTextList). Wenn trotz fehlendem Schreibrechts Veränderungen vorgenommen werden, so kann es vorkommen, dass die Text-Ids nicht mehr zu den Texten innerhalb eines Visualisierungselements passen.

Durch das Kommando „Visualisierungs-Text-Ids prüfen“ können solche Fehler innerhalb aller Visualisierungen ermittelt werden.

12.30.8 Nicht verwendete Textlisteneinträge entfernen

Symbol:

Dieser Befehl der Kategorie Textliste dient dazu, Texte innerhalb der GlobalTextList, die in keinem Visualisierungselement verwendet werden, zu entfernen.

12.31 Trace

Die Befehlskategorie 'Trace' enthält Befehle zum Arbeiten im Trace-Fenster im Online-Betrieb.

12.31.1 Befehle der Kategorie Trace

Das Plug-In 'TraceEditor' stellt Befehle der Kategorie 'Trace' für das Arbeiten im Trace-Editor bereit. Standardmäßig sind diese Befehle im Trace-Menü zu finden, wenn der Editor aktiv ist.

Wenn erforderlich, kann die Menüstruktur im 'Anpassen'-Dialog verändert werden.

Die folgenden Trace-Befehle sind verfügbar:

- Trace einloggen
- Start / Stop Trace
- Trigger zurücksetzen
- Cursor
- Scrollen
- Zoomen
- Anzeige zurücksetzen
- Komprimieren
- Dehnen
- Mehrfachkanal
- Trace laden...
- Trace speichern...

12.31.2 Trace einloggen

Mit diesem Befehl (Kategorie Trace) wird der für den Trace erzeugte Code explizit in die Steuerung geladen, um dort das Tracen zu aktivieren. Dies ist für die erste Verwendung der Trace-Funktionalität auf einer Applikation nötig, wie auch später immer dann, wenn die Trace-Konfiguration oder das Applikationsprogramm geändert wurden.

Beachten Sie: ein aktiver Trace wird durch eine grüne Hinterlegung des Trace-Objekts im Gerätebaum angezeigt.

12.31.3 Start/Stop Trace

Symbols: ,

Dieser Befehl (Kategorie Trace) startet bzw. stoppt die Darstellung der Trace-Aufzeichnung.

Wenn die Aufzeichnung gerade gestoppt ist, steht vor dem Befehl das Symbol und ein Ausführen des Befehls bedeutet, dass die Aufzeichnung wieder gestartet wird, also die Variablenwerte ab dem aktuellen Zeitpunkt wieder im Trace-Fenster dargestellt werden.

Wenn die Aufzeichnung gerade läuft, steht vor dem Befehl das Symbol und ein Ausführen des Befehls bedeutet, dass die Aufzeichnung gestoppt wird, d.h. im Trace-Fenster wird der gerade dargestellte Abschnitt der Variablenwertekurve(n) "eingefroren".

12.31.4 Trigger zurücksetzen

Symbol:

Dieser Befehl (Kategorie Trace) setzt die Anzeige der Trace-Aufzeichnung zurück auf die Darstellung der aktuellen Werte, nachdem sie durch ein Trigger-Ereignis unterbrochen wurde oder die Darstellung gestoppt worden war.

12.31.5 Cursor

Symbol:

Solange diese Option (Befehlskategorie Trace) aktiviert ist (Default), wird die aktuelle Position des Cursors im Koordinatensystem des Trace-Fensters immer in der Statusleiste des Programmiersystems angezeigt. Der Cursor kann bewegt werden, wenn mit der Maus in das schwarze Dreieck geklickt und dann bei gedrückter linker Maustaste gescrollt wird. Das Gleiche erreicht man, wenn man die **linke** bzw. **rechte Pfeiltaste** benutzt. Alle Tastaturkombinationen sind unter Trace Tastenkombinationen aufgelistet. Die Option wird automatisch deaktiviert, sobald eine der Optionen Scrollen oder Zoomen aktiviert ist.

12.31.6 Scrollen

Symbol:

Dieser Befehl (Kategorie Trace) aktiviert den Modus zum Scrollen des Cursors. In diesem Modus wird der Mauszeiger einen kurzen Moment lang als Doppelpfeil dargestellt und man kann die Zeitachse der gerade dargestellte Trace durch entsprechende Bewegung des Cursors im Trace-Fenster nach links oder rechts verschieben. Dasselbe kann mit **linker/rechter Pfeiltaste** bei gleichzeitig gedrückter **Alt Taste** erreicht werden. Für vertikales Scrollen kann die **nach oben/nach unten Pfeiltaste** verwendet werden. Wird zusätzlich die **Strg Taste** gedrückt, wird das horizontale und vertikale Scrollen in größeren Abständen durchgeführt..

Falls Mehrfachkanal angewählt ist, ist es möglich die Zeitachse aller Graphen gleichzeitig mit Maus oder per Tastatur (**linke/reche Pfeiltasten** bei gedrückter **Alt Taste**) zu scrollen. Zum vertikalen Scrollen des selektierten Graphen kann die **nach oben/nach unten Pfeiltaste** verwendet werden. Wird zusätzlich die **Ctrl Taste** gedrückt wird das horizontale und vertikale Scrollen in größeren Abständen durchgeführt.

Wenn außerhalb des im Puffer noch verfügbaren Wertebereichs gescrollt wird, erscheint die Angabe "Außerhalb des Anzeigebereichs" im Trace-Fenster.

Um den Scroll-Modus des Mauszeigers zu deaktivieren, aktivieren Sie den Cursor Modus.

Um die Standard-Anzeige wieder herzustellen, verwenden Sie den Befehl Anzeige zurücksetzen.

Um die Anzeige der Graphen zu zoomen verwende die Befehle zum Zoomen.

Alle Tastaturbefehle sind in Trace Tastenkombinationen aufgelistet.

12.31.7 Zoomen

Symbol:

Dieser Befehl (Kategorie Trace) aktiviert den Zoom-Modus für den Cursor. Wenn der Modus aktiviert ist, wird der Mauszeiger als dargestellt und man kann im Trace-Fenster ein Rechteck aufziehen, das den Bereich definiert, der so vergrößert werden soll, dass er das gesamte Trace-Fenster ausfüllt.

Um den Zoom-Modus zu deaktivieren, muss der Cursor-Modus aktiviert werden.

Um die Standard-Anzeige wieder herzustellen, verwenden Sie den Befehl Anzeige zurücksetzen

Beachten Sie auch die Möglichkeit, im Trace-Fenster mit Hilfe der **Scroll-Maus** oder über Tastatur zu zoomen:

- Drehen am Mausrad vergrößert/verkleinert das Koordinatensystem entlang der X- **und** Y-Achse. Das Gleiche kann im Ziffernblock über die **+** und **- Taste** erreicht werden.
- Drehen am Mausrad und gleichzeitiges Drücken der **Umsch Taste** zoomt nur entlang der X-Achse. Das Gleiche kann im Ziffernblock über die **+** und **- Taste** bei gleichzeitigem Drücken der **Umsch Taste** erreicht werden.
- Drehen am Mausrad und gleichzeitiges Drücken der **Strg Taste** zoomt nur entlang der Y-Achse. Das Gleiche kann im Ziffernblock über die **+** und **- Taste** bei gleichzeitigem Drücken der **Strg Taste** erreicht werden.

12.31.8 Anzeige zurücksetzen

Symbol:

Mit diesem Befehl (Kategorie Trace) kann die Anzeige der Graphen auf die Standardwerte zurückgesetzt werden, nachdem diese beispielsweise durch Zoomen verändert wurden. Die Standardeinstellungen sind im Konfigurations-Dialog definiert.

12.31.9 Dehnen

Symbol:

Mit diesem Kommando (Kategorie Trace) können die Werte der gesampelten Graphen gedehnt dargestellt werden. Mit weiterem Dehnen, das eins nach dem anderen angewendet wird, ist es möglich den Ausschnitt des angezeigten Traces im Fenster weiter zu verkleinern.

Dieses Kommando ist das Gegenstück zu Komprimieren.

12.31.10 Komprimieren

Symbol:

Mit diesem Kommando (Kategorie Trace) können die Werte der gesampelten Graphen komprimiert dargestellt werden. Mit weiterem Komprimieren, das eins nach dem anderen angewendet wird, ist es möglich den Ausschnitt des angezeigten Traces im Fenster weiter zu vergrößern.

Dieses Kommando ist das Gegenstück zu Dehnen.

12.31.11 Mehrfachkanal

Dieses Kommando (Kategorie Trace) ermöglicht es, die Ansicht der Trace-Graphen zu wechseln. Standardmäßig werden die Grahpen aller Variablen im selben Diagramm dargestellt. In Mehrfachkanal-Ansicht wird der Graph jeder Variablen in seinem eigenen Diagramm aber mit identischer X-Achse visualisiert. Zoomen und Scrollen beeinflusst die X-Achse aller Diagramme simultan.

Alle Tastaturbefehle sind in Trace Tastenkombinationen aufgelistet.

12.31.12 Trace laden...

Dieser Befehl (Kategorie Trace) wird verwendet um eine Trace-Konfiguration, die zuvor in eine Datei gespeichert wurde, wieder in den Editor zu laden. Die Datei kann mit Hilfe des Befehls Trace speichern... erzeugt worden sein.

Im Dialog **Trace laden** (Standarddialog zum Browsen im Dateisystem) kann nach einer Trace-Konfigurationsdatei gesucht werden. Der Dateityp ist auf "»PLC Designer« Trace" (*.trace)" gesetzt. Mit Beenden des Dialogs über **öffnen** wird die gespeicherte Konfiguration geladen und im Trace-Editor angezeigt.

12.31.13 Trace speichern...

Dieser Befehl (Kategorie Trace) wird verwende um eine aktuelle Trace-Konfiguration in einer Datei zu speichern. Der Befehl öffnet den Dialog **Trace speichern** (Standarddialog zum Browsen im Dateisystem), in dem eine Trace-Konfigurationsdatei gespeichert werden kann. Die Default-Erweiterung dieser Datei ist *.trace.

Die Konfiguration aus einer Trace-Datei kann mit Befehl Trace laden... wieder in den Editor geladen werden.

12.31.14 Trace Tastenkombinationen

Hinweis: Die Graphen des Trace-Diagramms ändern sich in größeren Abständen, wenn zusätzlich zu den folgenden Tastenkombinationen die **Strg-Taste** gedrückt wird.

Der Trace-Editor stellt folgende Tastenkombinationen standardmäßig zur Verfügung:

Tastenkombinationen	Aktion im Trace-Diagramm
links/rechts Pfeiltasten bei gedrückter **Alt Taste**	horizontales Scrollen der Zeitachse (X-Achse)
nach oben/nach unten Pfeiltasten (ohne gedrückter **Alt Taste**)	vertikales Scrollen der Zeitachse (X-Achse)
Taste + und **Taste -** (Ziffernblock oder Tastatur)	Zoomen des Koordinatensystems entlang der X- und Y-Achse
Taste + und **Taste -** bei gedrückter **Umschalttaste** (Ziffernblock oder Tastatur)	Zoomen entlang der X-Achse
Taste + und **Taste -** bei gedrückter **Strg Taste** (Ziffernblock oder Tastatur)	vertikales Zoomen entlang der Y-Achse
links/rechts Pfeiltasten	Scrollen des Cursors nach rechts bzw. links

Bei aktiviertem Mehrfachkanal erfolgen bei Gebrauch der Funktionstasten folgende Aktionen:

Tastenkombinationen	Aktion im Trace-Diagramm
links/rechts Pfeiltasten bei gedrückter **Alt Taste**	simultanes und horizontales Scrollen der x-Achse aller Diagramme
nach oben/nach unten Pfeiltasten (ohne gedrückter **Alt Taste**)	horizontales Scrollen nur des selektierten Diagramms
Taste + und **Taste -** (Ziffernblock oder Tastatur)	simultanes Zoomen des Koordinatensystems entlang der X- und Y-Achse aller Diagramme
Taste + und Taste - bei gedrückter **Umschalttaste** (Ziffernblock oder Tastatur)	horizontales Zoomen aller Diagramme entlang der X-Achse
Taste + und Taste - bei gedrückter **Strg Taste** (Ziffernblock oder Tastatur)	vertikales Zoomen aller Diagramme entlang der Y-Achse
links/rechts Pfeiltasten	Scrollen des Cursors nach rechts bzw. links

12.32 Rückgängig/Wiederherstellen

Die Befehlskategorie 'Rückgängig/Wiederherstellen' enthält Befehle zum Wiederherstellen vorheriger Aktionen in den Projekt-Editoren.

12.32.1 Befehle der Kategorie Rückgängig/Wiederherstellen

Das Plug-In 'Edit Commands' stellt Befehle für die Kategorie 'Rückgänig/Wiederherstellen' bereit, mit denen die vorangegangen Bearbeitungsschritte beim Editieren eines Objekts wiederhergestellt werden können.

- Rückgängig
- Wiederherstellen

Standardmäßig sind diese Befehle im 'Bearbeiten'-Menü zu finden. Wenn erforderlich, kann die Menüstruktur mit Hife des Anpassen-Dialogs umkonfiguriert werden.

12.32.2 Rückgängig

Symbol:

Standard Tastaturkürzel: <Strg> + <Z>

Dieser Befehl (Kategorie 'Rückgängig/Wiederherstellen') macht den zuletzt durchgeführten Bearbeitungsschritt im gerade aktiven Editor rückgängig.

Wiederholtes Ausführen des Befehls macht nach und nach alle Aktionen rückgängig, die seit Öffnen des Editorfensters durchgeführt wurden. Dies gilt für alle Aktionen in den Baustein-Editoren und im Geräte- bzw. POUs-Fenster.

Mit dem Wiederherstellen Befehl kann das Ergebnis eines gerade rückgängig gemachten Bearbeitungsschritts wieder hergestellt werden.

12.32.3 Wiederherstellen

Symbol:

Standard Tastaturkürzel: <Strg>+<Y>

Dieser Befehl (Kategorie 'Rückgängig/Wiederherstellen') stellt das Ergebnis eines gerade rückgängig gemachten Bearbeitungsschritts wieder her.

Wiederholtes Ausführen des Befehls macht stellt nach und nach alle Bearbeitungsschritte wieder her, die zuvor rückgängig gemacht wurden.

12.33 Benutzerverwaltung

Die Befehlskategorie 'Benutzerverwaltung' enthält Befehle zum Aufsetzen von Benutzerkonten und Benutzergruppen und für das Konfigurieren von Zugriffsrechten im Projekt.

12.33.1 Befehle der Kategorie Benutzerverwaltung

Mit dem Plug-In 'User Management UI' werden Befehle der Kategorie 'Benutzerverwaltung' installiert, für ...

- ... die Konfiguration von Zugriffsrechten auf die Projektobjekte.
- ... das Ein- /Ausloggen in ein bzw. aus einem Projekt über ein bestimmtes Benutzerkonto, um die diesem Konto zugewiesenen Zugriffsrechte auf das Projekt bzw. seine Objekte zu erhalten.

Die Konfiguration von Benutzerkonten und -gruppen wird im Unterdialog 'Benutzerverwaltung' der 'Projekteinstellungen' vorgenommen

Standardmäßig sind die folgenden Befehle Teil des Untermenüs 'Benutzerverwaltung' im Projekt-Menü enthalten:

- Benutzer anmelden...
- Benutzer abmelden
- Rechte....

Wenn erforderlich, kann die aktuelle Menükonfiguration im Anpassen-Dialog, Kategorie 'Menü' verändert werden.

12.33.2 Benutzer anmelden...

Symbol:

Dieser Befehl (Kategorie Benutzerverwaltung) öffnet den Dialog **Anmelden**, in dem den man sich über ein definiertes Benutzerkonto in einem Projekt oder einer Bibliothek anmelden kann.

Anmelden über ein Benutzerkonto bedeutet, daß man diejenigen Zugriffsrechte auf Projektobjekte erhält, die der Gruppe zugewiesen sind, zu der der Benutzer gehört. Die Konfiguration von Benutzerkonten und Gruppen wird im Unterdialog 'Benutzerverwaltung' der 'Projekteinstellungen' vorgenommen.

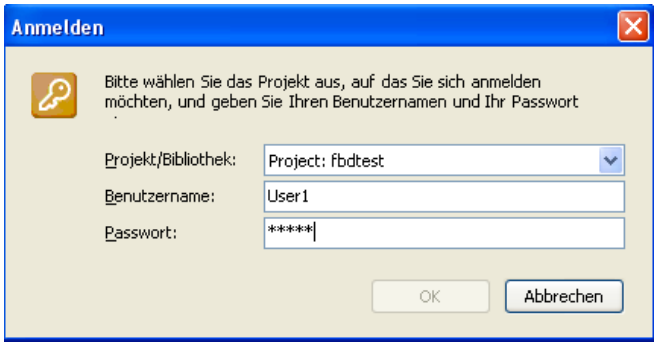


Abb. 270 Dialog 'Anmelden'

Zum Anmelden wählen Sie das gewünschte Projekt oder die eingebundene Bibliothek von der Auswahlliste im Feld **Projekt/Bibliothek**. Geben Sie **Benutzernamen** und **Passwort** eines gültigen Benutzerkontos ein. Beachten Sie, daß jedes Projekt und jede Bibliothek eine eigene Benutzer- und Zugriffsrechteverwaltung hat. Loggen Sie sich mit **OK** ein.

Wenn bereits ein anderer Benutzer angemeldet ist, wird dieser durch die neue Anmeldung automatisch abgemeldet.

Wenn Sie in einem Projekt oder einer Bibliothek angemeldet sind und versuchen, eine Aktion auszuführen, für die Sie kein Recht besitzen, wird automatisch der folgende **Anmelden-Dialog** geöffnet, der Ihnen ermöglicht, über ein anderes Benutzerkonto, das die entsprechenden Rechte gibt, einzuloggen:

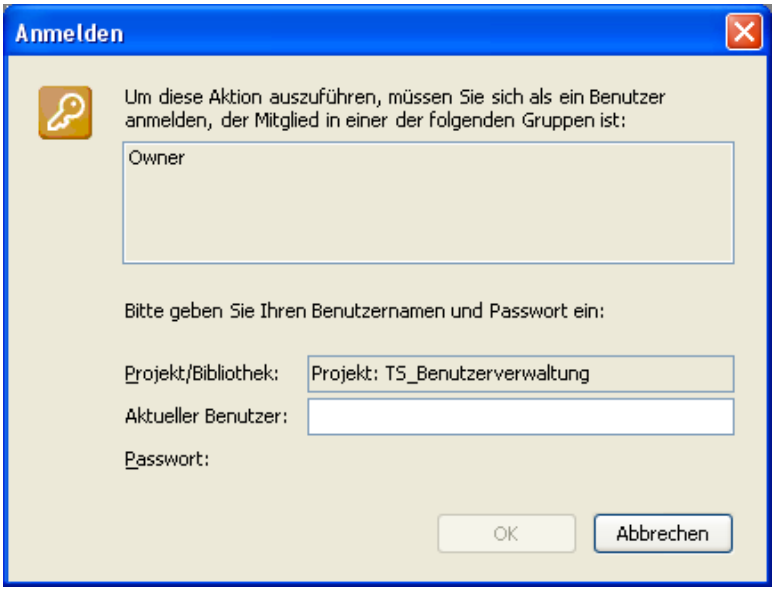


Abb. 271 Anmelden-Dialog im Falle einer nicht-erlaubten Aktion

Der obere Teil des Dialogs zeigt - nur zur Information - alle Gruppen, die die erforderlichen Rechte für die versuchte Aktion besitzen. Wenn Sie ein Benutzerkonto für eine dieser Gruppen verwenden können, können Sie nun die entsprechenden Zugangsdaten Benutzername und Passwort eingeben und dann die Aktion ausführen.

Die **Statusleiste** zeigt immer den aktuell im Projekt angemeldeten Benutzer.

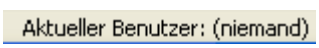


Abb. 272 Beispiel: Statusleiste

12.33.3 Benutzer abmelden

Symbol:

Dieser Befehl (Kategorie Benutzerverwaltung) meldet den gerade im Projekt oder der Bibliothek angemeldeten Benutzer wieder ab. Nur wenn gerade gar kein Benutzer eingeloggt ist, wird eine entsprechende Meldung ausgegeben, ansonsten erscheinen kein Dialog und keine Meldung.

Wenn der Benutzer gerade in mehreren Projekten oder eingebundenen Bibliotheken angemeldet ist (nicht unbedingt über das gleiche Benutzerkonto), wird der folgende **Abmelden-Dialog** geöffnet:

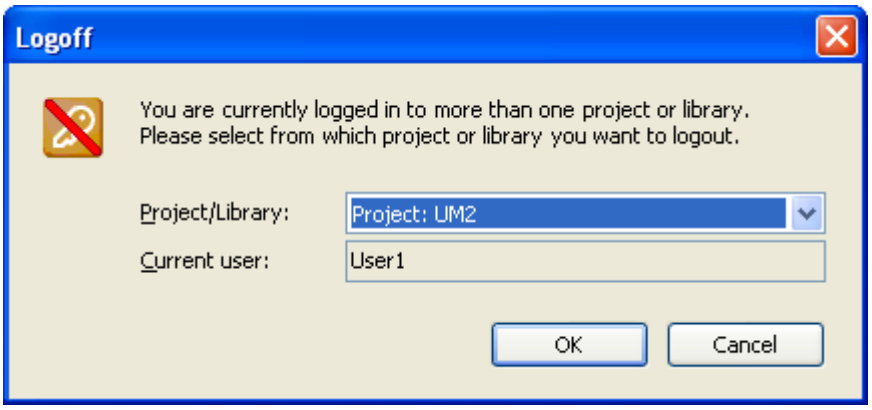


Abb. 273 Dialog 'Abmelden'

Hier kann nun über die Auswahlliste im Feld **Projekt/Bibliothek** konkret das Projekt bzw. die Bibliothek ausgewählt werden, aus dem/der der **Aktuelle Benutzer** abgemeldet werden soll.

Die **Statusleiste** zeigt immer den aktuell im Projekt angemeldeten Benutzer.

12.33.4 Rechte...

Dieser Befehl (Kategorie Benutzerverwaltung) öffnet den Dialog **Rechte**, in dem die Zugriffsrechte auf Objekte bzw. die Rechte, bestimmte Aktionen im Projekt auszuführen, konfiguriert werden können. Beachten Sie, daß die Zugriffsrechte auf Objekte auch im Objekt Eigenschaften Dialog gesetzt werden, mit identischer Wirkung.

Beachten Sie auch, dass alle Einstellungen im Rechte-Dialog sofort wirksam werden.

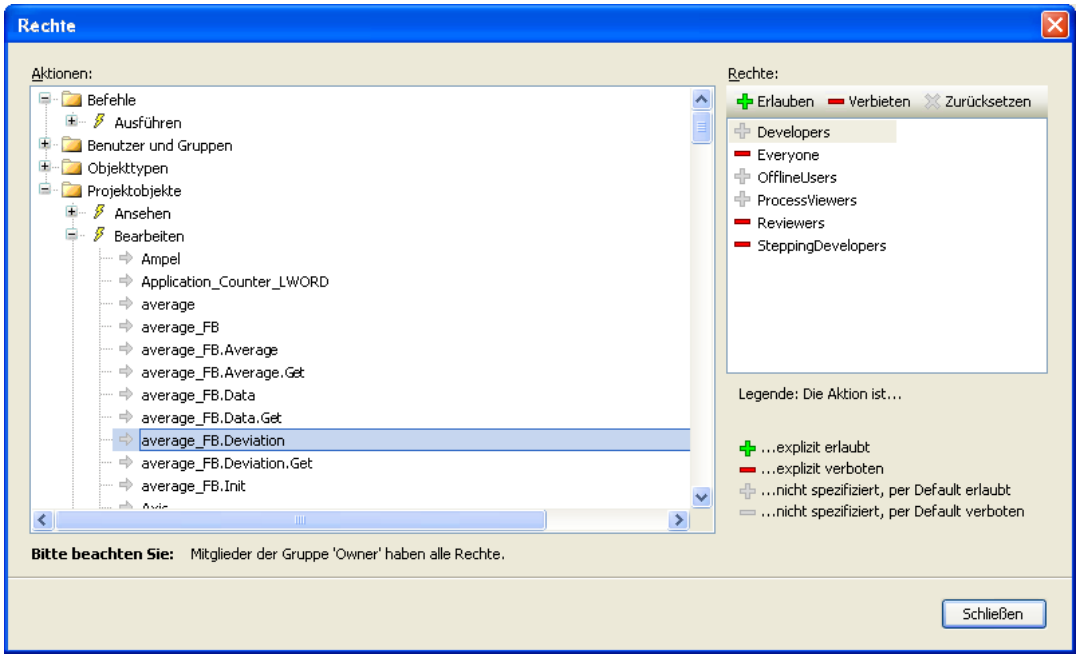


Abb. 274 Dialog 'Rechte'

Das Fenster **Aktionen** im linken Teil des Dialogs zeigt alle möglichen Rechte, d.h. alle Aktionen, die auf Objekte des Projekts ausgeführt werden können. Der Baum ist folgendermaßen strukturiert:

- Auf oberster Ebene: Namen einiger Kategorien, die verwendet werden, um den Baum zu strukturieren, und zwar nach Rechtevergabe für das Ausführen von **Befehle**n, das Bearbeiten von **Benutzer- und Gruppen**-Einstellungen, das Erzeugen von **Objekttypen** und das Ansehen, Bearbeiten, Entfernen und Handhaben von Kindobjekten von **Projektobjekte**n.
- Unterhalb dieser Kategorie-Einträge hängen die Einträge für die **Aktionen**, die jeweils ausgeführt werden können. Auch diese Knoten dienen nur der Strukturierung des Baums. Mögliche Aktionen:
  - Ausführen (eines Menübefehls)
  - Erzeugen (Anlegen eines neuen Objekts im Projekt)
  - Kindobjekte hinzufügen oder entfernen (zu bestehenden Objekten)
  - Bearbeiten (eines Objekts im Editor oder von Benutzer- und Gruppeneinstellungen im entsprechenden Dialog)

  - Entfernen (Löschen oder Ausschneiden eines Objekts)
  - Ansehen (Öffnen der Ansicht eines Objekts im Editor)
- Unterhalb jedes Aktionen-Knotens finden sich die möglichen "**Ziele**" der Aktion, z.B. die einzelnen Projektobjekte oder Menübefehle. Diese Einträge können konkret angewählt werden, um Rechte zu vergeben.

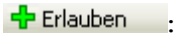
Das Fenster **Rechte** im rechten Teil des Dialogs zeigt eine Liste aller aktuell definierten Benutzergruppen (außer der Gruppe "Owner") und eine Symbolleiste zum Konfigurieren der Rechte für eine Gruppe.

Wählen Sie die Gruppe aus, für die Sie jetzt Rechte definieren wollen.

Links vom Gruppennamen zeigt eines der unten erklärten Symbole das Recht an, das gerade für das im Aktionen-Fenster ausgewählte "Ziel" einer Aktion gilt. Um ein Recht zu ändern, klicken Sie also auf den "Ziel"-Eintrag im Aktionen-Fenster. Die Rechte-Symbole:

- : Die gerade im Aktionen-Fenster aufgeklappte Aktion auf das/die gerade ausgewählte(n) Ziel(e) ist **erlaubt**, und zwar für die gerade selektierte Gruppe.
- : Die gerade im Aktionen-Fenster aufgeklappte Aktion auf das/die gerade ausgewählte(n) Ziel(e) ist **verboten**, und zwar für die gerade selektierte Gruppe.
- : Die gerade im Aktionen-Fenster aufgeklappte Aktion auf das/die gerade ausgewählte(n) Ziel(e) wurde nicht explizit erlaubt, aber sie ist **per Default erlaubt**, z.B. weil sie für das "Vaterobjekt" erlaubt wurde. Beispiel: Die Gruppe bekam dieses Recht für Objekt "myplc", somit ist es auch Standardrecht für Objekt "myplc.pb_1". "Erlaubt" ist die Standardeinstellung für alle Rechte, die nicht explizit konfiguriert wurden.
- : Die gerade im Aktionen-Fenster aufgeklappte Aktion auf das/die gerade ausgewählte(n) Ziel(e) wurde nicht explizit verboten, aber sie ist **per Default verboten**, z.B. weil sie für das "Vaterobjekt" verboten wurde.
- Wenn gerade mehrere Aktionen im Aktionen-Fenster ausgewählt sind, für die die Gruppe nicht ein- und dasselbe Recht besitzt, wird kein Symbol angezeigt.

Um die Rechte für eine Gruppe zu ändern, wählen Sie die entsprechende Aktion und das/die Ziel(e) im Aktionen-Fenster und verwenden die entsprechende Schaltfläche in der Symbolleiste des Rechte-Fenster:



: Explizites Erlauben.

: Explizites Verbieten.

: Das Recht wird auf den Standardwert zurückgesetzt.

12.34 Visualisierungskommandos

Die Befehlskategorie 'Visualisierungskommandos' enthält Befehle für das Arbeiten im Visualisierungs-Editor, z.B. für das Anordnen der Visualisierungselemente.

12.34.1 Befehle der Kategorie Visualisierungskommandos

Das Plug-In 'Visual Editor' stellt Befehle für die Befehlskategorie 'Visualisierungskommandos' bereit, die für das Arbeiten im Visualisierungs-Editor bereitstehen.

Standardmäßig sind die Befehle im Menü 'Visualisierung' bzw. im Kontextmenü zu finden. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs verändert werden.

Die Befehle:

- Visualisierungselement einfügen
- Frame Auswahl
- Schnittstellen-Editor
- Tastaturbedienung
- Elementliste
- Hintergrund

Auswahl:

- Alles selektieren
- Alles deselektieren
- Gruppieren
- Gruppierung aufheben

Ausrichtung:

- Links ausrichten
- Oben ausrichten
- Rechts ausrichten
- Unten ausrichten
- Horizontal zentrieren
- Vertikal zentrieren

Reihenfolge:

- Um Eins nach vorn legen
- Nach vorn legen
- Nach hinten legen
- Um Eins nach hinten legen
- Tastaturbedienung aktivieren

12.34.2 Visualisierungselement einfügen

Dieser Befehl (Kategorie Visualsierungskommandos) dient dem Einfügen eines Visualisierungselements im Visualisierungs-Editor. Dies entspricht dem Einfügen per Drag&Drop aus dem 'Werkzeuge'-Fenster (Toolbox).

Der Befehl öffnet ein Untermenü, in dem aus den aktuell verfügbaren Elementen das gewünschte ausgewählt werden. Das Element wird daraufhin in der linken oberen Ecke des Editor-Fensters eingefügt.

12.34.3 Alles selektieren

Mit diesem Befehl (Kategorie Visualsierungskommandos) werden alle Visualisierungselemente der gerade im Editor bearbeiteten Visualisierung ausgewählt

Siehe auch: Alles deselektieren

12.34.4 Alles deselektieren

Mit diesem Befehl (Kategorie Visualsierungskommandos) wird die aktuelle Selektion von Visualisierungselementen aufgehoben.

Siehe auch: Alles selektieren

12.34.5 Gruppieren

Symbol:

Mit diesem Befehl (Kategorie Visualsierungskommandos) werden die gerade ausgewählten Visualisierungselemente gruppiert und die Gruppe als Einzelobjekt selektiert dargestellt. Für die Mehrfachauswahl halten Sie die <Umschalt>-Taste gedrückt während Sie die gewünschten Elemente anklicken. Alternativ können Sie außerhalb eines Elements ins Editorfenster klicken und bei gedrückter Maustaste ein Rechteck um die gewünschten Elemente ziehen.

Zum Auflösen der Gruppe verwenden Sie Befehl Gruppierung aufheben.

Die folgende Abbildung zeigt die Gruppierung (von links nach rechts) bzw. das Aufheben der Gruppierung (von rechts nach links) zweier Rechteck-Elemente:

12.34.6 Gruppierung aufheben

Symbol:

Mit diesem Befehl (Kategorie Visualsierungskommandos) wird eine Gruppe von Visualisierungselementen wieder aufgelöst. Die einzelnen Elemente werden jeweils wieder einzeln selektiert dargestellt.

Eine Gruppe kann mit Befehl Gruppieren gebildet werden.

12.34.7 Frame Auswahl

Dieser Befehl (Kategorie Visualsierungskommandos) dient dem Konfigurieren eines **Frame** Elements.

Ein Frame Element wird verwendet, um eine Untereinheit einer Visualisierung zu definieren. Diese Untereinheit enthält eine oder mehrere andere Visualisierungen, von denen immer eine im Online-Betrieb dargestellt wird, wobei zwischen den verfügbaren Visualisierungen hin- und hergeschaltet werden kann.

Grundsätzlich wird die erste aus der Liste der dem Frame zugewiesenen Visualisierungen dargestellt. Aber der Anwender kann durch eine Eingabe auf ein anderes Visualisierungselement, das entsprechend konfiguriert wurde (Eingabe-Eigenschaften, Frame-Visualisierung umschalten), bewirken, dass eine der anderen Visualisierungen im Frame gezeigt wird. Somit ist ein Wechseln zwischen verschiedenen Darstellungen innerhalb einer Visualisierung möglich, im Gegensatz zu »PLC Designer« V2.3, wo komplett zwischen den einzelnen Visualisierungen hin- und hergesprungen werden mußte.

Die Visualisierungen in einem Frame sind Referenzen, d.h. Instanzen der originalen Visualisierungen, und Platzhalter, die in der Original-Visualisierung definiert sind, können im Frame durch lokal passende Werte ersetzt werden. Sehen sie in diesem Zusammenhang: Frames,_Referenzen,_Schnittstelle (Platzhalter)

Die Auswahl der Visualisierungen in einem Frame wird im **Frame Konfiguration** - Dialog vorgenommen, der über den 'Frame Auswahl' Befehl geöffnet wird:

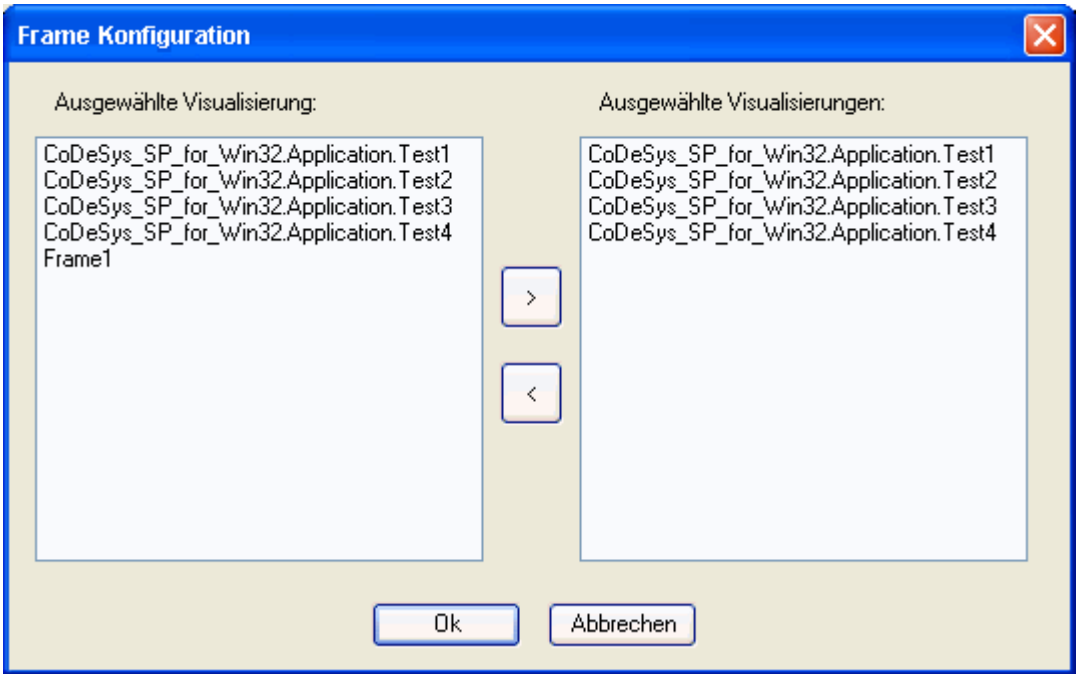


Abb. 275 Dialog 'Frame Konfiguration'

Auf der linken Seite sehen Sie die im Projekt **Verfügbaren Visualisierungen**. Wählen Sie diejenigen aus, die im Frame referenziert werden sollen. Über die Pfeil-Schaltflächen können Sie in die Liste der **Ausgewählten Visualisierungen** auf der rechten Seite des Dialogs eingefügt werden bzw. wieder daraus entfernt.

Es wird empfohlen, einem Frame nur Visualisierungen aus dem globalen Pool zuzuweisen. Ansonsten könnte es zu Problemen kommen, falls später Geräte- oder Applikationsobjekt umbenannt werden und dann der Pfad der zugewiesenen Visualisierungen nicht mehr gültig ist.

Die Reihenfolge der ausgewählten Visualisierungen von oben nach unten bestimmt die automatisch erzeugten impliziten Index-Nummern für die Visualisierungen. Die oberste erhält "0", die folgenden "1", "2" usw.. Die Index-Nummern werden für die Konfiguration der Umschaltefunktion (Frame-Visualisierung umschalten) eines anderen Elements benötigt. Initial wird die Visualisierung mit Index "0" dargestellt.

Beispiel:

Sie möchten eine Visualisierung, die eine Schaltflächenleiste enthält und ein Anzeigefeld, in dem im Online Modus je nachdem welche Schaltfläche betätigt wird, eine andere Visualisierung angezeigt wird. Legen Sie also eine Visualisierung visu_xy und drei weitere Visualisierungen visu1, visu2, visu3, die später in visu_xy angezeigt werden sollen. Fügen Sie drei Rechtecke (Schaltflächen) und einen Frame in visu_xy ein. Konfigurieren Sie das Frame-Element mit der 'Frame Auswahl'-Funktion und weisen ihm die Visualisierungen visu1, visu2 und visu3 zu. Konfigurieren Sie die Schaltflächen jeweils so, dass bei einem Mausklick darauf visu1, visu2 bzw. visu3 im Frame aufgerufen wird (Eingabekonfiguration, Eigenschaft OnMouseClick, Frame-Visualisierung umschalten).

12.34.8 Schnittstellen-Editor

Symbol:

Standard Tastenkürzelt: <Alt> + <F6>

Dieser Befehl (Kategorie Visualsierungskommandos) öffnet den Schnittstellen-Editor zum Definieren von "Platzhalter Variablen" in einer Visualisierung, die dann im Frame-Element einer anderen Visualisierung eingefügt, d.h. **referenziert** werden soll.

12.34.9 Tastaturbedienung

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) öffnet den Konfigurationseditor für die Tastaturbedienung für die aktuelle Visualisierung. Er wird in einer Registerblattansicht im oberen Teil des Visualisierungs-Editors angezeigt.

12.34.10 Elementliste

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) öffnet den Elementlisten-Editor für die aktuelle Visualisierung. Er wird in einer Registerblattansicht im oberen Teil des Visualisierungs-Editors angezeigt.

12.34.11 Hintergrund

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) öffnet den Dialog 'HIntergrund', in dem eine Farbe und/oder eine Abbildung für den Visualisierungshintergrund ausgewählt werden kann. Aktivieren Sie die gewünschte(n) Option(en):

Bitmap: Um ein Hintergrundbild zu definieren, muss hier der Pfad einer Bilddatei, die in einer Bildersammlung im Projekt verfügbar ist, eingetragen werden. Geben Sie dazu den Namen der Bildersammlung und die ID der Bilddatei - getrennt durch einen Punkt "." - ein: <bildersammlung>.<ID> (z.B. "Images_1.drive_icon>", "Images_1.43").

Farbe: Um die Hintergrundfarbe der Visualisierung zu definieren, wählen Sie aus der Farbauswahlliste die gewünschte Farbe aus.

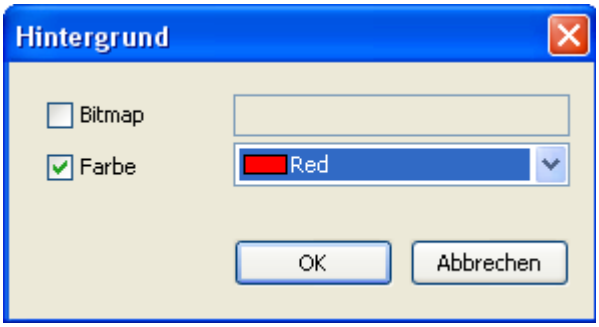


Abb. 276 Dialog 'Hintergrund'

12.34.12 Reihenfolge

Um Eins nach vorn legen

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) positioniert das ausgewählte Element eine Ebene höher, d.h. weiter in den Vordergrund der Visualisierung. Elemente auf tieferen Ebenen werden von denen auf höheren verdeckt.

Nach vorn legen

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) positioniert das ausgewählte Element im Vordergrund der Visualisierung, d.h. auf "höchster" Ebene. Elemente auf tieferen Ebenen werden von denen auf höheren verdeckt.

Um Eins nach hinten legen

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) positioniert das ausgewählte Element eine Ebene tiefer, d.h. weiter in den Hintergrund der Visualisierung. Elemente auf tieferen Ebenen werden von denen auf höheren verdeckt.

Nach hinten legen

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) positioniert das ausgewählte Element im Hintergrund der Visualisierung, d.h. auf "tiefster" Ebene. Elemente auf tieferen Ebenen werden von denen auf höheren verdeckt.

12.34.13 Ausrichtung

Links ausrichten

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) bewirkt, dass alle ausgewählten Visualisierungselemente am linken Rand ihres am weitesten links positionierten Elements ausgerichtet werden.

Oben ausrichten

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) bewirkt, dass alle ausgewählten Visualisierungselemente am oberen Rand ihres am weitesten oben positionierten Elements ausgerichtet werden.

Rechts ausrichten

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) bewirkt, dass alle ausgewählten Visualisierungselemente am rechten Rand ihres am weitesten rechts positionierten Elements ausgerichtet werden.

Unten ausrichten

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) bewirkt, dass alle ausgewählten Visualisierungselemente am unteren Rand ihres am weitesten unten positionierten Elements ausgerichtet werden.

Horizontal zentrieren

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) bewirkt, dass alle ausgewählten Visualisierungselemente an ihrer gemeinsamen horizontalen Mitte ausgerichtet werden.

Vertikal zentrieren

Symbol:

Dieser Befehl (Kategorie Visualsierungskommandos) bewirkt, dass alle ausgewählten Visualisierungselemente an ihrer gemeinsamen vertikalen Mitte ausgerichtet werden.

12.34.14 Tastaturbedienung aktivieren

Symbol:

Dieser Befehl (Kategorie Visual Commands) ist in der Menüleiste für eine integrierte Visualisierung verfügbar. Er aktiviert bzw. deaktiviert die Tastaturbedienung im Online-Modus einer Visuailsierung.

Wenn die Tastaturbedienung aktiviert ist, können Eingaben auf Elemente und die Auswahl der Elemente über bestimmte Tastenkombinationen vorgenommen werden. In diesem Fall werden andere über Tastatur gegebene Befehle nicht ausgeführt, solange der Visualisierungseditor aktiv und im Online-Modus ist.

12.35 Ansicht

Die Befehlskategorie "Ansicht" enthält Befehle zum Aufrufen bestimmter Ansichten-Fenster. Beachten Sie zusätzlich auch die Befehlskategorie 'Meldungsfenster'.

12.35.1 Querverweisliste

Befehle der Kategorie Ansicht - Querverweisliste

Das Plug-In 'Cross Reference View' stellt den Befehl Querverweisliste für Befehlskategorie Ansicht bereit, der dazu dient, die Referenzen einer Variable im Projekt als "Querverweisliste" in einem Fenster darzustellen.

Standardmäßig ist der Befehl im Ansicht-Menü enthalten. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog verändert werden.

Querverweisliste

Dieser Befehl (Kategorie Ansicht) öffnet ein Fenster, in dem die Querverweise für eine bestimmte Projektvariable aufgelistet werden. Als Querverweise gelten alle Positionen im Projekt oder begrenzt nur die innerhalb eines Bausteins (POU), an denen die Variable verwendet wird.

Die Ansicht heißt **Querverweisliste**.

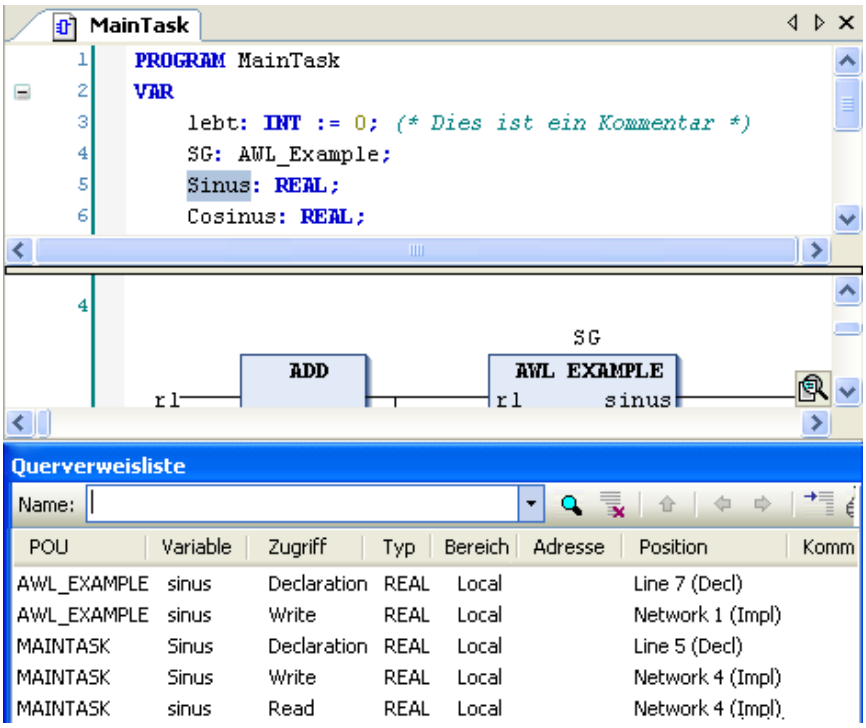


Abb. 277 Querverweisliste, Beispiel

Wenn **alle Querverweise** innerhalb des gesamten Projekts aufgelistet werden sollen: Tippen Sie den Variablenbezeichner manuell ein oder kopieren Sie ihn mit Kopieren&Einfügen aus einem Editorfenster ins Feld **Name** und bestätigen mit der <Eingabe>-Taste oder klicken auf die Schaltfläche .

Wenn **nur die Querverweise innerhalb der gleichen POU** aufgelistet werden sollen: Markieren Sie entweder den Variablenbezeichner in einem Editorfenster und ziehenihn mit der Maus in das Querverweislistenfenster, oder aktivieren Sie die Option Querverweise automatisch bei Selektionsänderung aktualisieren und platzieren den Mauszeiger im Bezeichnernamen im Editorfenster bzw. selektieren Sie diesen. In diesem Fall wird der Bezeichner automatisch ins Feld **Name** übernommen und die Querverweise darunter aufgelistet.

Wenn ein gültiger Bezeichner eingegeben wurde, werden die gefundenen Verwendungsstellen in Tabellenform mit folgenden Informationen zur Variable dargestellt:

POU	Name des Bauteins, in dem die Variable verwendet wird.
Variable	Variablenname
Zugriff	Wie wird die Variable an der gegebenen Position verwendet bzw. wie wird auf sie zugegriffen: Deklaration / Lesen / Schreiben / Aufruf
Typ	Datentyp
Bereich	Gültigkeitsbereich: Global / Lkcal
Adresse	IEC-Adresse, falls definiert
Position	Position der Variablenverwendung innerhalb des Editors (z.B. Zeilennummer, Netzwerknummer)
Kommentar	Kommentar, der ggfs. bei der Deklaration der Variable hinzugefügt wurde.

Die Liste kann pro Spalte alphabetisch **sortiert** werden. Durch einen Mausklick auf den Spaltentitel wird die Sortierung zwischen auf- und absteigender Anordnung hin und her geschaltet.

Ein Doppelklick in eine Zeile der Querverweisliste **öffnet die betroffene POU** und die genannte Position wird ausgewählt. Dies entspricht der Verwendung von

Schaltfläche ("**Zeige Position**"), wenn ein Eintrag in der Liste ausgewählt ist.

Schaltfläche **("Zeige vorige Position"**, Tastaturkürzel: <Umsch> + <F4>) bewirkt, dass zum vorangehenden Eintrag in der Liste gesprungen wird.

Schaltfläche (**"Zeige nächste Position"**, Tastaturkürzel: <F4>) bewirkt, dass zum nächsten Eintrag in der Liste gesprungen wird.

Schaltfläche (**"Gehe zur Definition"**, Tastaturkürzel: <F2>) bewirkt, dass an die Stelle gesprungen wird, an der die jeweilige Variable deklariert wird. Dazu wird der entsprechende Deklarationseditor geöffnet und die Variable dort markiert.

Schaltfläche entspricht dem Befehl Querverweise ausgeben und bewirkt eine **Ausgabe der aktuellen Querverweisliste ins Meldungsfenster**. Das kann nützlich sein, wenn die aktuelle Liste bereitgehalten werden soll, obwohl die automatische Aktualisierung aktiviert ist und sich somit die Liste im Querverweislisten-Fenster ändern kann.

12.35.2 Haltepunkte

Befehle der Kategorie Ansicht - Haltepunkte

Das Plug-In 'Breakpoints' stellt Befehle für Befehlskategorie 'Ansicht' bereit, die der Anzeige des Haltepunkte-Dialogs und der Aufrufliste dienen.

Standardmäßig sind diese Befehle im Ansicht-Menü enthalten. Wenn erforderlich, können Sie die Menükonfiguration im Anpassen-Dialog verändern.

Die Befehle:

- Haltepunkte
- Aufrufliste

Haltepunkte (Dialog)

Symbol:

Dieser Befehl (Kategorie 'Ansicht') öffnet den **Haltepunkte**-Dialog, der eine Übersicht über alle aktuell im Projekt gesetzten Haltepunkte (Breakpoints) gibt. Die Haltepunkte-Parameter werden dargestellt und können verändert werden. Außerdem können Haltepunkte hinzugefügt und entfernt werden bzw.aktiviert oder deaktiviert.

Also Haltepunkt-Parameter werden zunächst diejenigen angezeigt, die beim Setzen des Breakpoints mit Befehl Haltepunkt umschalten" (hierbei werden für die Haltepunkt-"Bedingungen" Default-Werte gesetzt) oder mit Befehl "Neuer Haltepunkt" (hierbei können die Bedingungen direkt definiert werden).

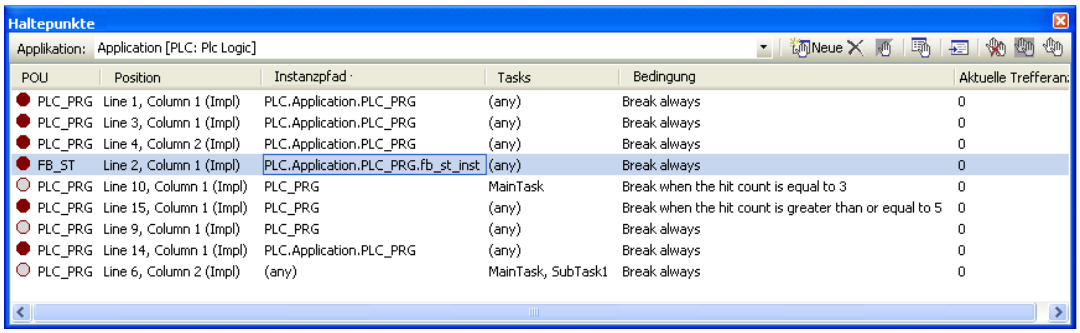


Abb. 278 Haltepunkte-Dialog

Applikation	Name der aktiven Applikation	Beispiel: "App1 [PLC:Plc Logic]"
POU	Name des Bausteins, der den Haltepunkt enthält	Beispiel: "PLC_PRG"
Position	Haltepunkt-Position innerhalb der POU: Zeilen- plus Spaltennummer (Texteditor) oder Netzwerk bzw. Elementnummern (grafischer Editor); "(Impl)" im Fall von Funktionsblöcken zeigt an, dass der Haltepunkt in der Implementation des Funktionsblocks sitzt, nicht in einer Instanz.	Beispiel: "Line 2, Column 1 (Impl)"
Instanzpfad	Vollständiger Objektpfad der Haltepunkt-Position	Beispiel: "PLC.Application.PLC_PRG.FBinst1"
Tasks	Namen der Tasks, bei deren Ausführung der Haltepunkt wirksam sein soll. Wenn keine Einschränkung gilt (Default), steht hier "(alle)".	Beispiel:: "MainTask, SubTask1"
Bedingung	Angabe, wann (in welcher Abhängigkeit von der Trefferanzahl) der Haltepunkt wirksam werden soll; mögliche Werte siehe Neuer Haltepunkt (Default: 1).	Beispiel: "Break when the hit count is equal to 3"
Aktuelle Trefferanzahl	Angabe, wie oft der Haltepunkt während der Ausführung bis jetzt bereits durchlaufen ("getroffen") wurde.	Beispiel: "3"

Die folgenden Funktionen sind über **Schaltflächen** im oberen rechten Teil des Dialogs verfügbar, um die Haltepunkt-Parameter des in der Liste ausgewählten Haltepunkts zu bearbeiten oder um Haltepunkte neu hinzuzufügen oder zu löschen:

	Neuer Haltepunkt	Öffnet den Dialog 'Neuer Haltepunkt'. Sehen Sie hierzu die Beschreibung des entsprechenden Befehls 'Neuer Haltepunkt'.
	Haltepunkt löschen	Entfernt den Haltepunkt; Achtung: nicht mit "Deaktivieren" verwechseln!
	Haltepunkt aktivieren/deaktivieren	Schaltet den Haltepunkt zwischen Status 'aktiviert' ● und 'deaktiviert' ○ hin und her; Im Falle des Deaktivierens wird der Haltepunkt nicht aus der Liste entfernt und kann wieder aktiviert werden.
	Eigenschaften	Dialog 'Haltepunkteigenschaften' wird geöffnet, wo die Haltepunkt-Parameter bearbeitet werden können. Der Dialog entspricht dem "Neuer Haltepunkt"-Dialog.
	Gehe zur Quellcodeposition	Der Dialog 'Online-Status anwählen' wird geöffnet, von wo man zur Position des Haltepunkts im Quellcode gelangen kann.

	Alle Haltepunkte löschen	Löscht alle Haltepunkte der Applikation. Die Liste wird geleert. Achtung: Nicht mit Deaktivieren verwechseln!
	Alle Haltepunkte aktivieren	Aktiviert ( ) alle gerade deaktivierten Haltepunkte.
	Alle Haltepunkte deaktivieren	Deaktiviert ( ) alle gerade aktivierten Haltepunkte. Die Haltepunkte bleiben in der Liste und können wieder deaktiviert werden.

Aufrufliste

Symbol:

Dieser Befehl (Kategorie 'Ansicht') öffnet das Fenster mit der **Aufrufliste**. Beim schrittweisen Ausführen eines Programms wird hier immer die aktuell erreichte Position mit vollständigem Aufruf-Pfad angezeigt.

Das Fenster zeigt unterhalb der Titelleiste immer den Namen der aktiven **Applikation** und den Namen der **Task**, die die gerade erreichte POU kontrolliert.

Die Aufrufliste besteht aus einer Liste von Positionen, jeweils beschrieben durch den **POU-**Namen, die **Position** und - im Fall von Instanzen - den **Instanzpfad**. Abhängig vom Editor wird die Position durch Zeilen- und Spaltennummer (Texteditor) oder durch Netzwerk- oder Elementnummern (grafischer Editor) angegeben.

Die erste Zeile in der Liste, mit einem gelben Pfeil markiert, beschreibt die aktuelle Ausführungsposition. Wenn diese Position in einem Baustein liegt, der von einem anderen aufgerufen wird, wird die Position des Aufrufs in der zweiten Zeile beschrieben. Wenn der Aufrufer wiederum von einem anderen Baustein aufgerufen wird, wird diese Aufrufposition in der dritten Zeile beschrieben usw.

Die Aufrufliste ist auch im Offline-Modus verfügbar bzw. im normalen Online-Betrieb (ohne gerade Debugging-Funktionen zu benützen). In diesem Fall enthält sie die zuletzt während eine schrittweisen Ausführung angezeigte Position, allerdings in "gegrauter" Schrift.

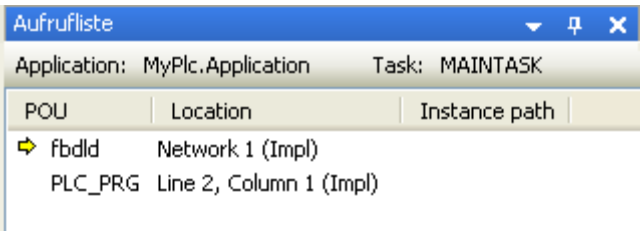


Abb. 279 Aufrufliste, Aktuelle Position fbdId", aufgerufen von "PLC_PRG"

12.35.3 Standardansichten

Befehle der Kategorie Ansicht - Standardansichten

Das Plug-In 'Default Navigators' stellt Befehle für die Befehlskategorie 'Ansicht' bereit, mit denen Fenster für Standardansichten in der Benutzeroberfläche geöffnet werden können:

- POUs
- Geräte

Standardmäßig enthalten diese Ansichten in der oberen rechten Ecke jeweils eine Schaltfläche zum Öffnen eines Menüs mit den folgenden Befehlen:

Aktion ausführen: Öffnet den im Fenster ausgewählten Baustein im entsprechenden Editor.

Objekt suchen: Der Dialog 'Objekt suchen' öffnet, wo der Objektbaum im Fenster nach einem bestimmten Objekt durchsucht werden kann. Geben Sie bei **Suchen nach:** eine Zeichenfolge, z.B. genau den gesuchten Namen oder auch nur ein einzelnes Zeichen ein und aktivieren Sie die gewünschten Suchoptionen:

- Ganzes Wort suchen: Alle Objektnamen, die genau dem Such-String entsprechen, werden gesucht.
- Groß-/Kleinschreibung: Wenn diese Option aktiviert ist, werden nur alle Objektnamen, die dem Such-String auch bezüglich Groß-/Kleinschreibung exakt entsprechen, gesucht.
- Präfix suchen: Alle Objektnamen, die mit dem gegebenen Such-String beginnen, werden gesucht.
- Teilstring suchen: Alle Objektnamen, die den gegebenen Such-String enthalten, werden gesucht.

Die gefundenen Objekte werden mit **Name** und **Pfad** im **Ergebnis** -Fenster aufgelistet.

Sortieren nach Typ, **Sortieren nach Name**: Die Objekte im Fenster werden nach Typ oder Name und alphabetisch sortiert.

Aufsteigend sortieren, **Absteigend sortieren**: Die oben gewählte Sortierung wird in aufsteigender oder absteigender Reihenfolge ausgeführt.

POUs Ansicht

Symbol:

Im POUs Fenster werden Programmiereinheiten des Projekts (Steuerungsprogramms) organisiert, die dann für die Verwendung in einer Applikation instanziert werden können.

Eine Beschreibung der Symbolleiste finden Sie unter Befehle der Kategorie Ansicht - Standardansichten.

Geräte Ansicht

Symbol:

Im Geräte-Fenster werden alle für das Projekt benötigten Geräte konfiguriert und die Applikationen entsprechend definiert.

Eine Beschreibung der Symbolleiste finden Sie unter Befehle der Kategorie Ansicht - Standardansichten.

12.35.4 PLC Designer Frame

Befehle der Kategorie Ansicht - »PLC Designer« Frame

Das »PLC Designer« Frame Plug-In stellt Befehle der Kategorie 'Ansicht' bereit, die dazu dienen, den Eigenschaften-Dialog für ein Objekt zu öffnen bzw. für die Benutzeroberfläche den Vollbildmodus zu aktivieren und zu deaktivieren.

Standardmäßig sind diese Befehle im Menü 'Ansicht' enthalten. Wenn erforderlich, kann mit Hilfe des Anpassen-Dialogs die Menüstruktur umkonfiguriert werden.

Die Befehle:

- Ganzer Bildschirm
- Eigenschaften...
- Startseite

Eigenschaften...

Symbol:

Dieser Befehl (Kategorie 'Ansicht') öffnet den Dialog 'Eigenschaften <Objektname>'. Die Eigenschaften des gerade im POUs- oder Gerätebaum ausgewählten Objekts werden auf verschiedenen Registerblättern angezeigt, abhängig vom Typ des Objekts. Die Einstellungen können in diesem Dialog auch geändert werden:

- Allgemein (Objektname, Objekttyp, Standard-Editor)
- Bootapplikation
- Bitmap
- Dynamische Speichereinstellungen
- Externe Datei
- Visualisierung (Visualisierung als Dialog oder (Ziffer-)Tastatur verwenden, Darstellungsgröße, Hintergrundbild berücksichtigen)
- Zugriffskontrolle (Rechte für Benutzergruppen)
- Quellcode-Verwaltung (Datenbankanbindung, Checkout-Status)
- Übersetzen (von Übersetzen ausschließen, Externe Implementierung, Systemaufrufe ermöglichen, Immer binden, Compiler-Definitionen)
- Netzwerkvariablen (für Globale Variablenlisten)
- Netzwerkeinstellungen (für Globale Netzwerkvariablenlisten)
- AS-Einstellungen (Flags für Ablaufsprache)
- Verknüpfung mit Datei (für Export und Import von Globalen Variablenlisten)

Allgemein

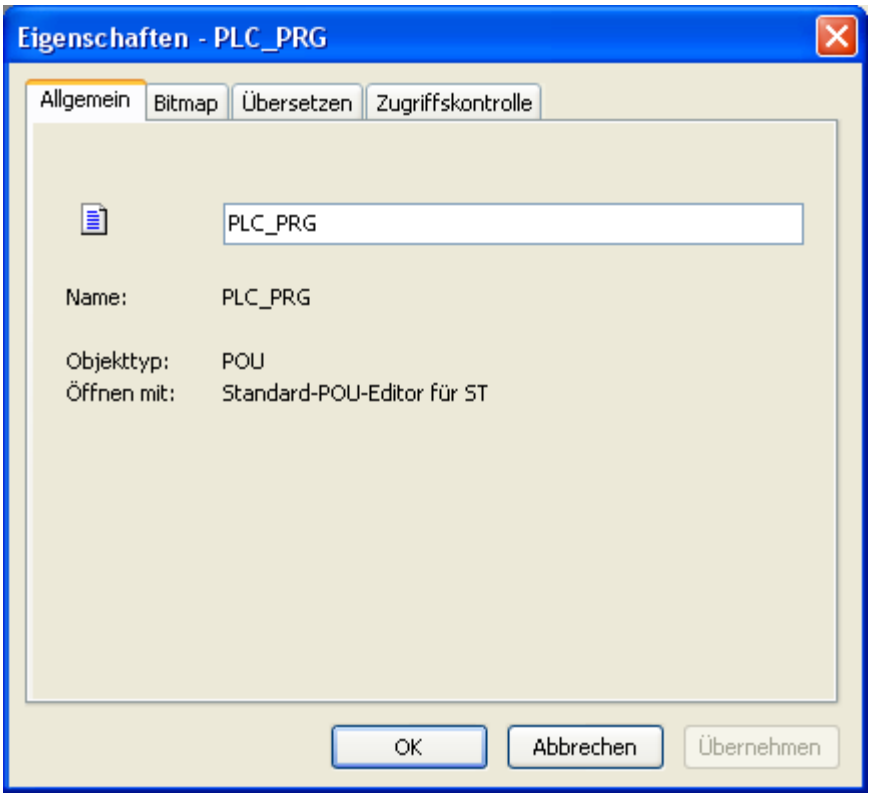


Abb. 280 Eigenschaften-Dialog, Kategorie Allgemein

Generelle Informationen zum Objekt:

Name: Objektname, wie in POUs- oder Gerätefenster

Objekttyp: Art des Objekts (z.B. "POU", "Applikation", "Inteface" etc.)

Öffnen mit: Typ des Editors, in dem das Objekt angezeigt bzw. bearbeitet wird.

Bootapplikation

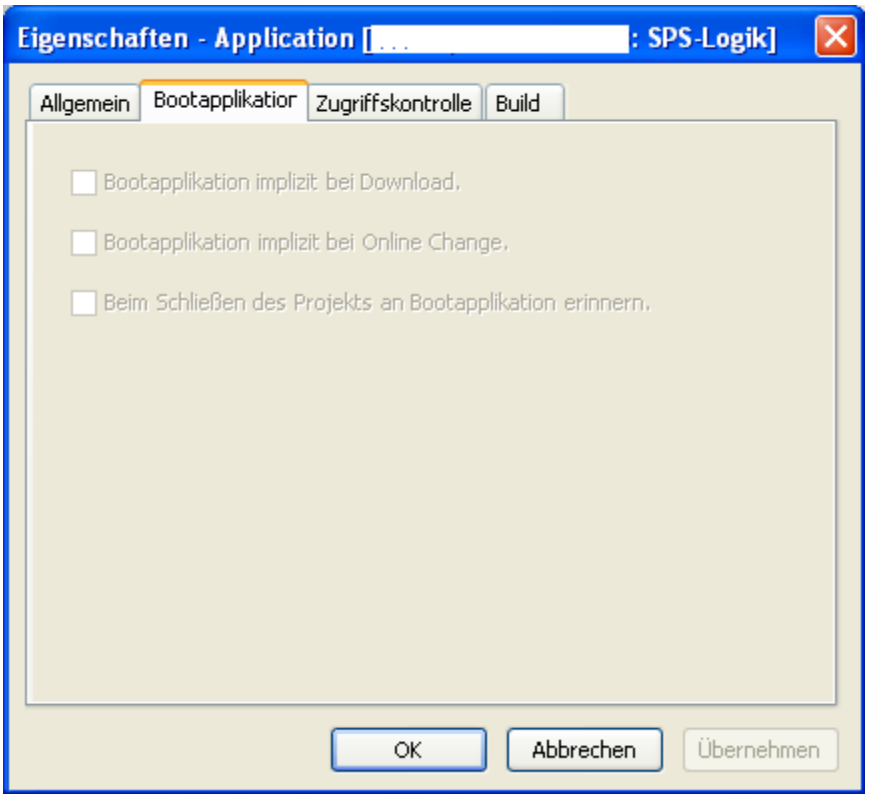


Abb. 281 Eigenschaften-Dialog, Kategorie Bootapplikation

Es hängt vom Gerät ab, ob diese Einstellungen verfügbar sind:

Bootapplikation implizit bei Download: Wenn aktiviert, wird beim Projekt-Download automatisch eine Bootapplikation erzeugt.

Bootapplikation implizit bei Online Change: Wenn aktiviert, wird bei Online Change automatisch eine Bootapplikation erzeugt.

Beim Schließen des Projekts an Bootapplikation erinnern: Wenn aktiviert, wird vor dem Verlassen des Projekts nachgefragt, ob die Bootapplikation (neu) angelegt werden soll.

Bitmap

Hier kann dem Objekt eine Bilddatei zugewiesen werden bzw. von ihm entfernt werden, mit dem es in der grafischen Ansicht des Bibliotheksverwalters und in der Toolbox des FUP/IKOP/IL-Editors dargestellt wird. Transparenz der Abbildung kann durch Auswahl einer Farbe erreicht werden, die dann transparent dargestellt wird. Dazu muss die Option **Transparenzfarbe** aktiviert werden, dann kann über die rechteckige Schaltfläche rechts davon der Standarddialog zur Auswahl einer Farbe geöffnet werden.

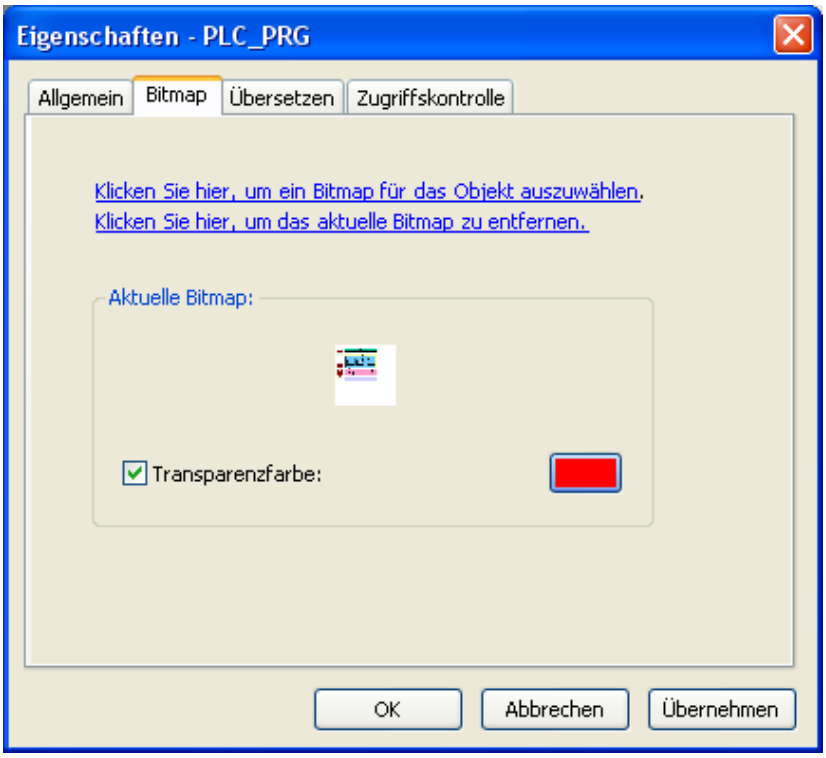


Abb. 282 Eigenschaften-Dialog, Kategorie Bitmap

Dynamische Speichereinstellungen

Abb. 283 Eigenschaften-Dialog, Kategorie Dynamische Speichereinstellungen

Dynamische Speicherallozierung verwenden: Aktivieren Sie diese Option, wenn für die Applikation automatisch Speicher reserviert werden soll. In diesem Fall geben Sie die gewünschte **Maximale Speichergröße (bytes)** an. Zu beachten ist, dass nicht der gesamte Speicher für die dynamische Objekterzeugung bereit steht, sondern ein Teil davon für Verwaltungsinformationen verwendet wird.

Die aktuellen Einstellungen werden bei jedem Download oder Online Change zur Steuerung übertragen.

Externe Datei

Hier werden die Einstellungen angezeigt, die Sie beim Einfügen einer externen Datei mit Hilfe des 'Objekt hinzufügen'-Dialogs vorgenommen haben. Die Einstellungen können hier geändert werden.

Um die Einstellungen wirksam werden zu lassen, ohne den Dialog zu schließen, drücken Sie Schaltfläche **Anwenden**.

Visualisierung

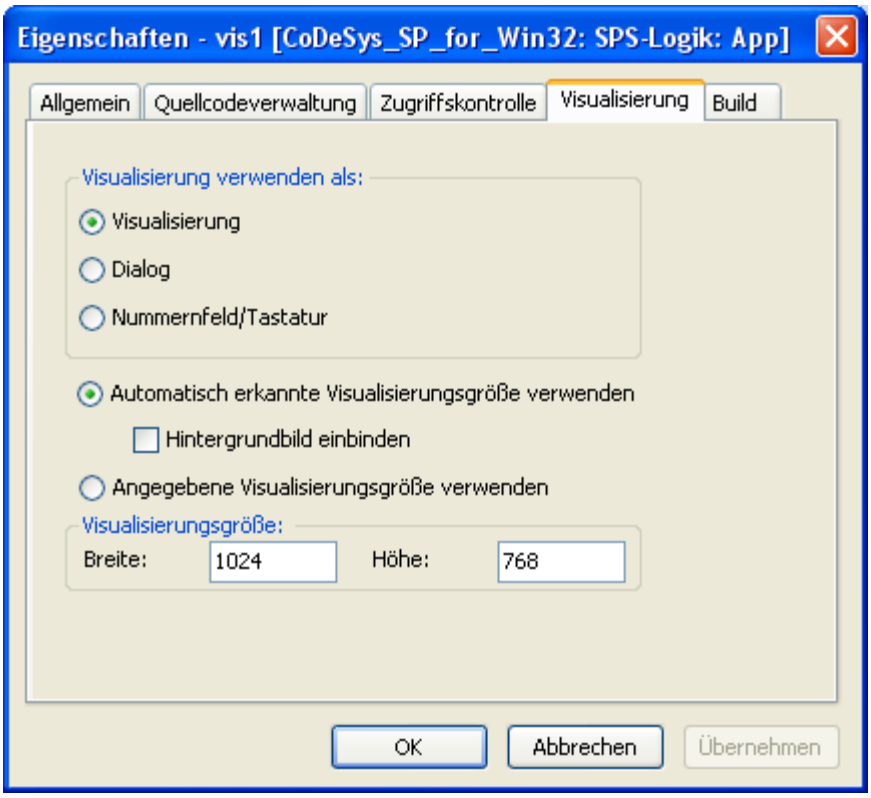


Abb. 284 Eigenschaften-Dialog, Kategorie Visualisierung

Für die gerade ausgewählte Visualisierung werden folgende Einstellungen angezeigt und können verändert werden:

Visualisierungsgröße: aktuelle **Breite** und **Höhe** der Visualisierung (Anzahl Pixel)

Automatisch ermittelte Visualisierungsgröße verwenden: Die Größe der Visualisierung wird ermittelt, bei der alle gerade enthaltenen Visualisierungselemente und das Hintergrundbild sichtbar sind. Wenn die Option **Hintergrundbild einschließen** jedoch deaktiviert ist, wird das Hintergrundbild - falls es den Bereich, der alle Visualisierungselemente einschließt, überragt - abgeschnitten.

Angegebene Visualisierungsgröße verwenden: Die Visualisierungsgröße wird durch die oben angezeigten Werte definiert (Breite, Höhe), wobei nicht berücksichtigt wird, ob alle Visualisierungselemente und das Hintergrundbild in diesen Bereich passen, d.h. vollständig sichtbar sind.

Beachten Sie, dass im Visualization Manager angegeben werden kann, ob die Visualisierungsgröße, die hier definiert ist, automatisch der Bildschirmgröße des Visualisierungs-Clients (Target-Visualisierung) angepaßt werden soll, oder nicht.

Zugriffskontrolle

Dieser Dialog ist nur verfügbar, wenn die Plug-Ins für die Benutzerverwaltung im aktuellen Profil enthalten sind. Er ermöglicht, für die vorhandenen Benutzergruppen Zugriffsrechte auf das Objekt zu konfigurieren. Dies entspricht der Konfiguration im 'Rechte'-Dialog, der standardmäßig über das Menü Benutzerverwaltung geöffnet werden kann.

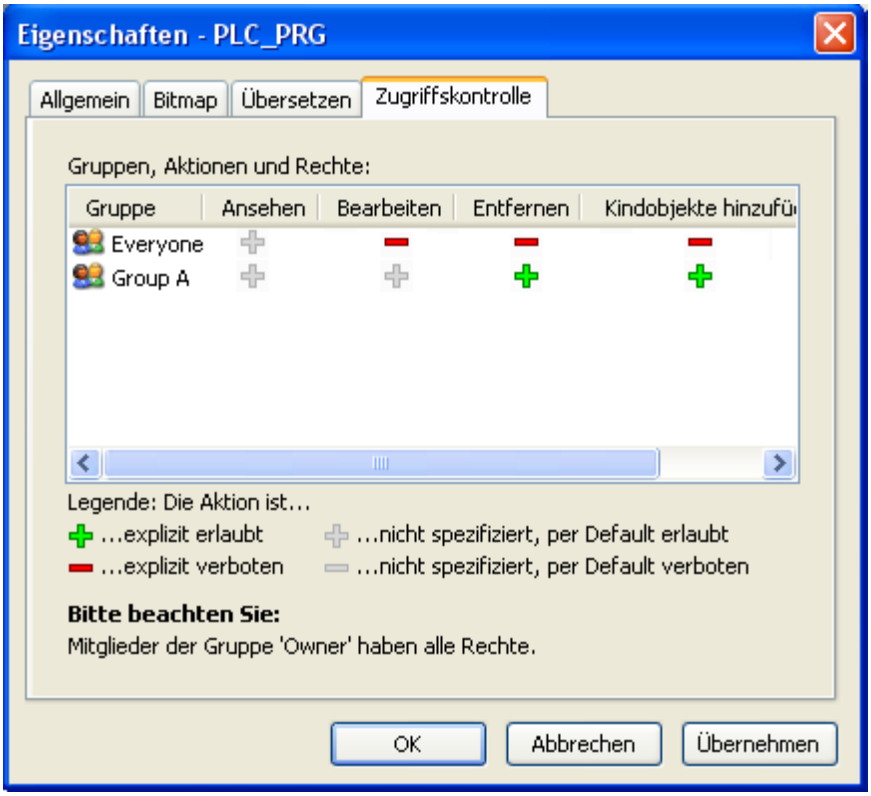


Abb. 285 Eigenschaften-Dialog, Kategorie Zugriffskontrolle

Um ein Recht für eine bestimmte Aktion und Gruppe umzusetzen, wählen Sie das entsprechende Feld in der Tabelle aus, öffnen über <Leertaste> oder einen Mausklick die Auswahlliste und wählen das gewünschte Recht aus.

Eine Beschreibung der möglichen Aktionen, Rechte und der verwendeten Symbole finden Sie auf der Hilfeseite zum Rechte-Dialog.

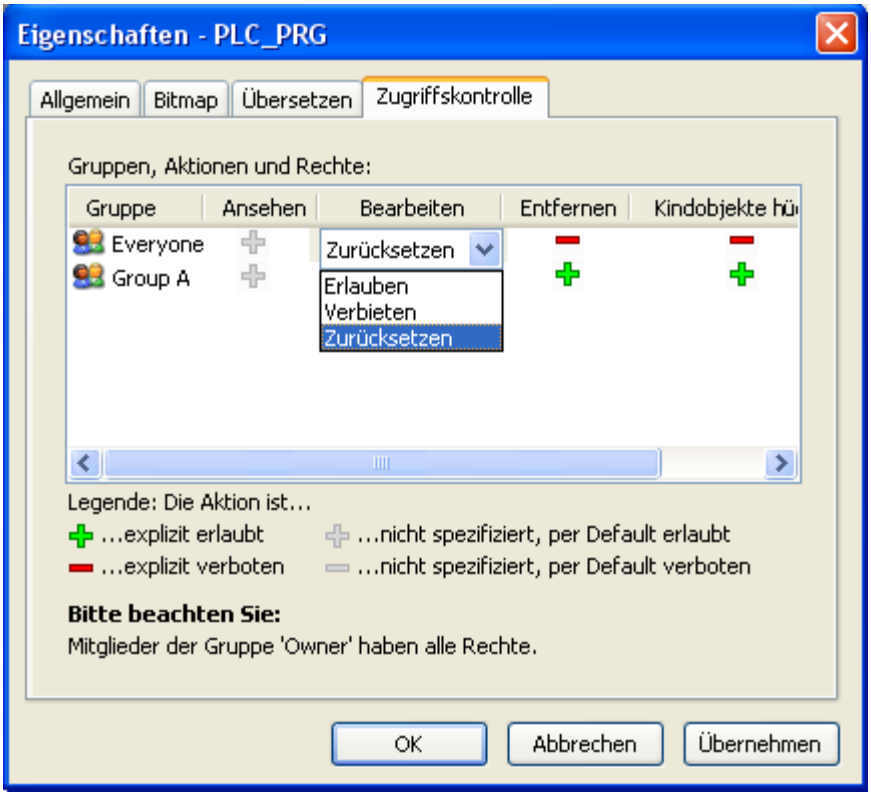


Abb. 286 Beispiel: Auswahlliste der Rechte für die Aktion 'Bearbeiten' für Gruppe 'Everyone'

Quellcode-Verwaltung

Dieser Dialog ist nur verfügbar, wenn das Projekt der Quellcodeverwaltung unterliegt, d.h. wenn die Objekte des Projekts über ENI in einer externen Datenbank verwaltet werden.

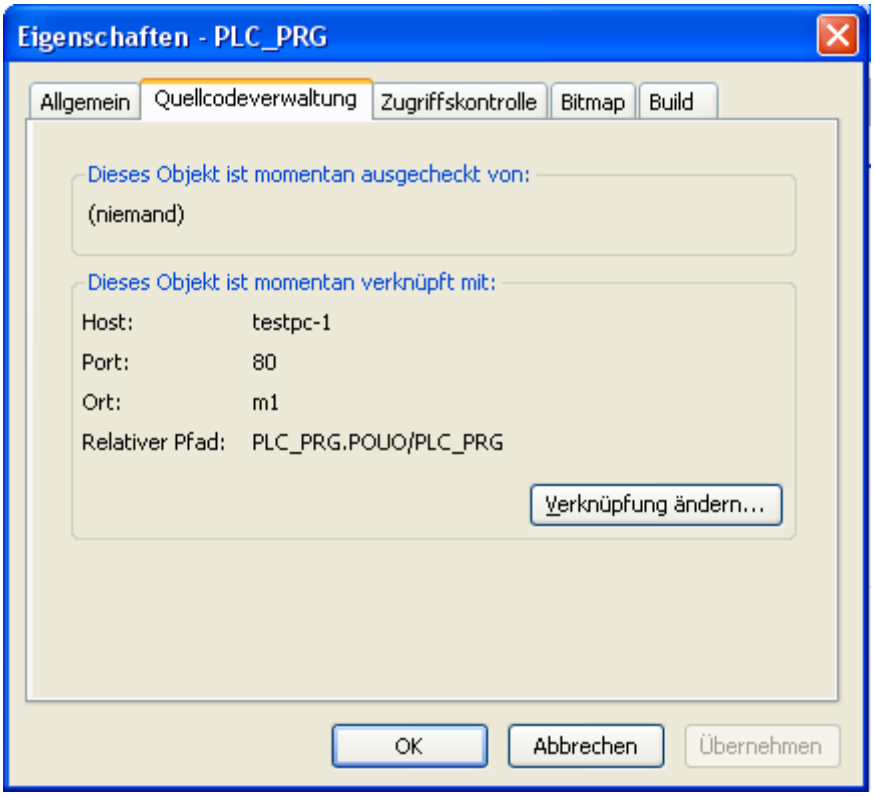


Abb. 287 Eigenschaften-Dialog, Kategorie Quellcodeverwaltung

Dieses Objekt ist momentan ausgecheckt von: Hier ist zu sehen, ob das Objekt gerade ausgecheckt ist und von welchem Benutzer.

Dieses Objekt ist momentan verknüpft mit: Hier wird der Datenbankort angegeben, an den das Objekt gebunden ist:

Host: Name des Rechners, auf dem die Datenbank liegt.

Port: Nummer des Ports, der für die Verbindung zur Datenbank verwendet wird.

Ort: Datenbank-Ordner, der das Objekt enthält.

Relativer Pfad: Pfad des Objekts in der Datenbank, relativ zur oben angegebenen location.

Verknüpfung ändern: Diese Schaltfläche öffnet den Dialog 'Change Source Control Binding', in dem Sie den Datenbankpfad, an den das Objekt gebunden ist, verändern können:

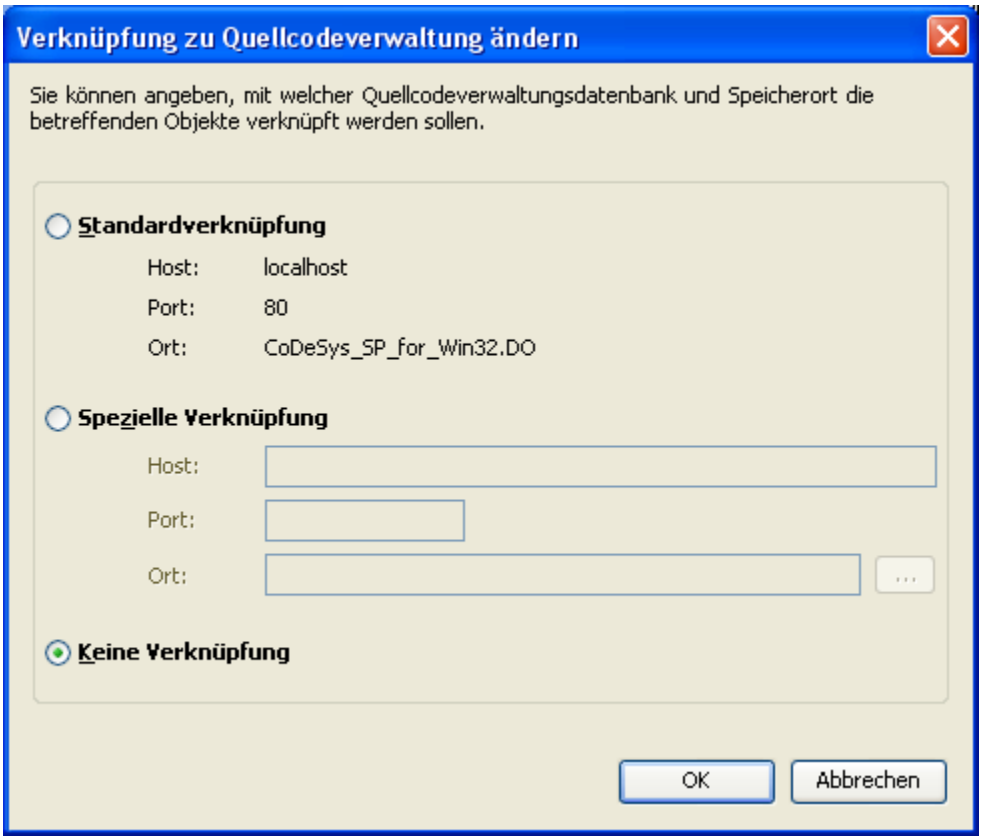


Abb. 288 Eigenschaften-Dialog, Quellcodeverwaltung ändern

Standardverknüpfung: Diese Option ist standardmäßig aktiviert.. Das Objekt ist an den Datenbankort gebunden, der definiert wurde, als das Projekt über den Dialog Quellcodeverwaltung ändern in Quellcodeverwaltung gebracht wurde. Die entsprechenden Angaben zu Host, Port und Datenbankort werden angezeigt.

Spezielle Verknüpfung: Um das Objekt an einen anderen als den Default-Datenbankort (s.o.) zu binden, aktivieren Sie diese Option und geben den entsprechenden Host-Namen, die Port-Nummer und den Datenbankort ein. Zur Eingabe des Ortes kann die Schaltfläche [...] verwendet werden, um den Standarddialog zum Browsen nach einem Ordner zu erhalten.

Keine Verknüpfung: Wenn diese Option aktiviert ist, wird das Objekt nicht in der Datenbank verwaltet. Es wird dann nur lokal im Projekt gehalten (entsprechend der in »PLC Designer« V2.3 verwendeten Objektkategorie "Lokal").

Übersetzen

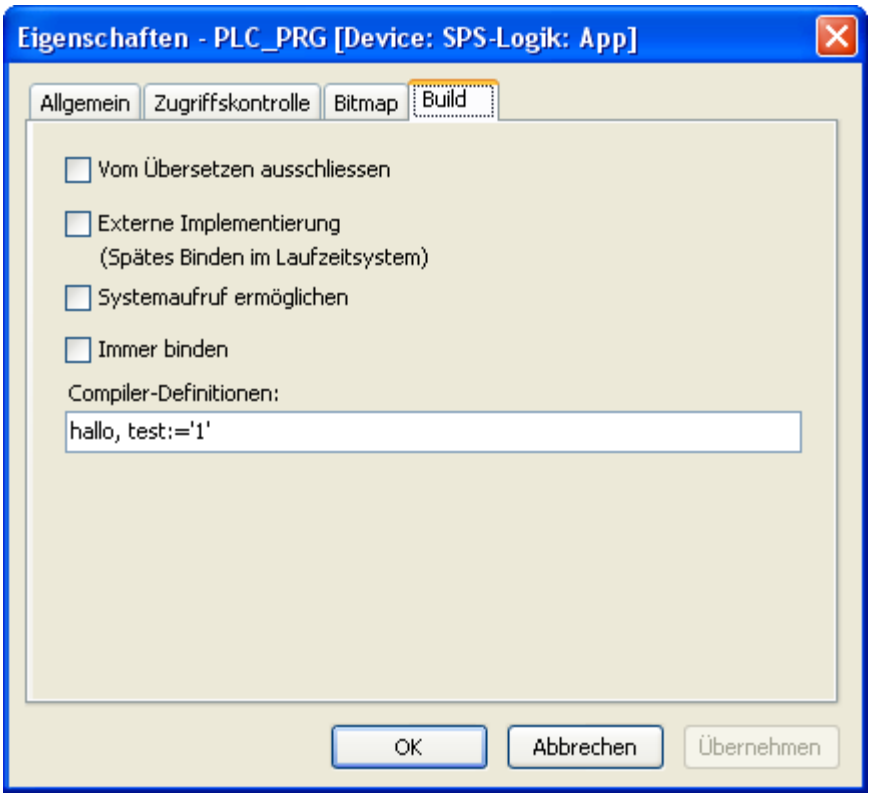


Abb. 289 Eigenschaften-Dialog, Kategorie Übersetzen

Hier können Optionen bezüglich der Kompilation (Übersetzen) des Objekts aktiviert werden:

Vom Übersetzen ausschließen: Das Objekt wird beim nächsten Übersetzungslauf nicht berücksichtigt.

Externe Implementierung (Spätes Binden im Laufzeitsystem): Beim Übersetzen des Projekts wird für dieses Objekt kein Code generiert. Das Objekt wird erst gelinkt, wenn das Projekt auf dem Zielsystem läuft, vorausgesetzt, es ist dort vorhanden (z. B. in einer Bibliothek).

Systemaufrufe ermöglichen: Hintergrund: Im Gegensatz zu »PLC Designer« V2.3 kann jetzt der ADR-Operator mit Funktions-, Programm-, Funktionsblock- und Methodennamen verwendet werden und ersetzt den INSTANCE_OF Operator. Sehen Sie in diesem Zusammenhang 'Funktionszeiger'. ABER es gibt keine Möglichkeit, Funktionszeiger innerhalb vom »PLC Designer« aufzurufen. Um einen Systemaufruf (Laufzeitsystem) zu ermöglichen, muss für das Funktionsobjekt die Option 'Systemaufrufe ermöglichen' aktiviert werden.

Immer binden: Das Objekt ist beim Compiler markiert und damit immer in der Compile-Information enthalten. Dies bedeutet, dass es immer kompiliert und auf die SPS geladen wird. Diese Option wird dann relevant, wenn das Objekt unterhalb einer Applikation liegt oder über Bibliotheken referenziert wird, die ebenfalls unterhalb einer Applikation liegen. Die Compile-Information wird ebenfalls als Basis für die auswählbaren Variablen der Symbolkonfiguration verwendet.

Compiler-Definitionen: Hier können "defines" (siehe {define} Anweisungen) und Bedingungen für die Kompilation des Objekts eingetragen werden (bedingte Kompilierung). Der Ausdruck *expr*, der in solchen Pragmas verwendet wird, kann auch hier eingegeben werden, und mehrere Einträge sind in Form einer komma-separierten Liste möglich.

Beispielsweise kann gewünscht sein, die Kompilierung einer Applikation vom Wert einer bestimmten Variable abhängig zu machen. (Siehe ein Beispiel in der folgenden Abbildung: "hallo" ist definiert und wenn die Variable 'test' den Wert "2" hat, wird die Applikation übersetzt.)

Netzwerkvariablen

Wenn die Netzwerkfunktionalität vom Gerät unterstützt wird, können die Netzwerkeigenschaften einer Globalen Variablenliste (GVL)im Eigenschaften-Dialog angezeigt und bearbeitet werden.

Netzwerkeigenschaften für eine GVL zu definieren bedeutet, die in der GVL deklarierten Variablen als Netzwerkvariablen zur Verfügung zu stellen. Eine GVL muss zu diesem Zweck vom "Sender" der Netzwerkvariablen bereitgestellt werden. Der Empfänger muss als Gegenstück eine entsprechende Globale Netzwerkvariablenliste definieren, die die gleichen Variablendeklarationen enthält.

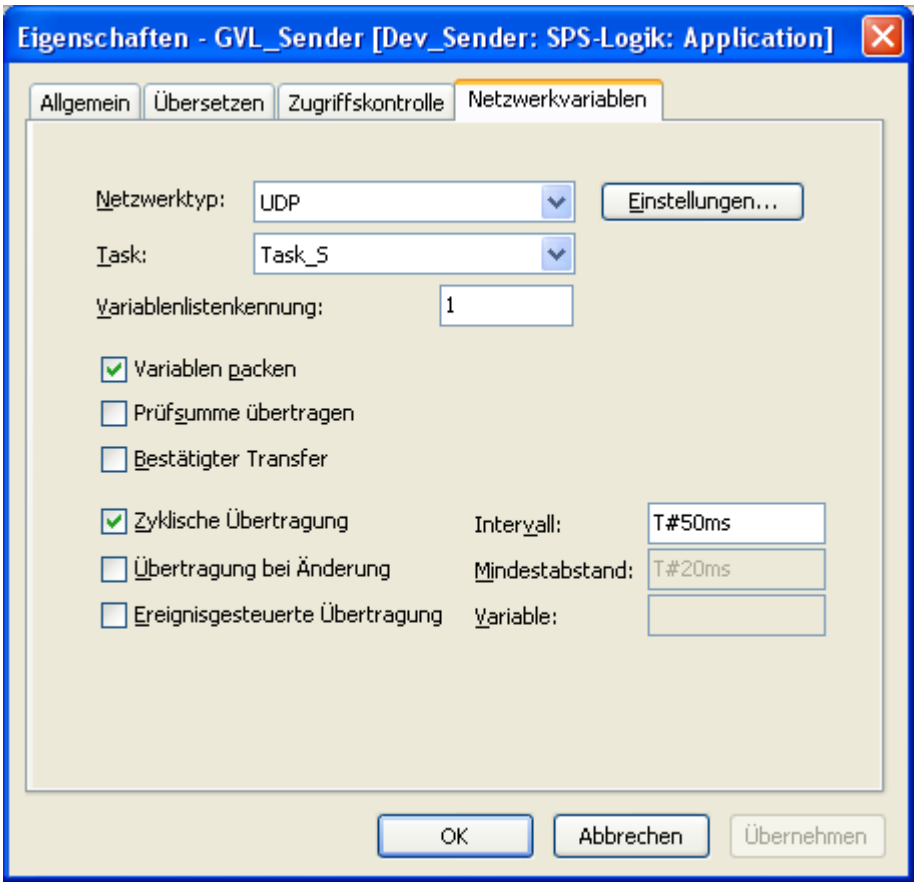


Abb. 290 Eigenschaften-Dialog, Kategorie Netzwerkvariablen (GVL)

Netzwerktyp: Wählen Sie den gewünschten Typ von der zielsystemabhängigen Liste. Beispielsweise CAN" für ein CAN-Netzwerk, "UDP" für ein UDP-System.

Task: Wählen Sie von der Auswahlliste die Task der aktuellen Applikation, die die zu sendenden Variablen kontrolliert. Die Variablen werden immer am Ende eines Taskzyklus versendet.

Variablenlistenkennung: Nummer der Liste, die als erste gesendet werden soll (Default=1). Weitere Listen werden mit aufsteigender Nummerierung gekennzeichnet.

Einstellungen: Protokollspezifische Einstellungen; die möglichen Einträge hängen von der jeweiligen Netzwerkbibliothek ab.

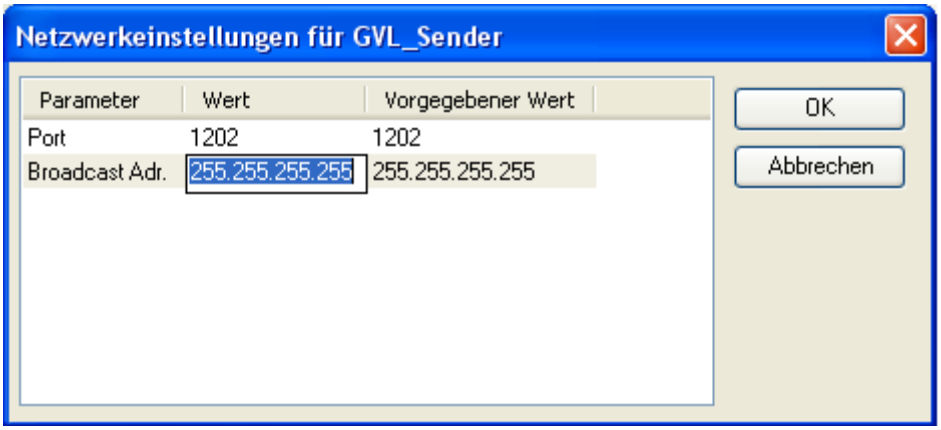


Abb. 291 Netzwerkeinstellungen

Für "UDP"-Netzwerke müssen die folgenden **Parameter** gesetzt werden:

Port: Nummer des Ports, der für den Datenaustausch mit anderen Netzwerkteilnehmernverwendet wird; der **Vorgegebene Wert** ist "1202"; der aktuelle Wert kann jederzeit im Feld **Wert** verändert werden (wählen Sie das Feld aus und drücken <Leertaste> um in den Eingabemodus zu gelangen); Stellen Sie sicher, dass die anderen Knoten im Netzwerk denselben Port definieren! Wenn mehr als eine UDP-Verbindung im Projekt definiert ist, wird die Port-Nummer automatisch in allen Konfigurationen an den hier gesetzten Wert angepasst.

Broadcast Adr.: Der **Vorgegebene Wert** ist "255 . 255 . 255 . 255", was bedeutet, dass der Datenaustausch mit allen Netzwerkteilnehmern erfolgen wird. Der aktuelle Wert kann im Feld **Wert** verändert werden (wählen Sie das Feld aus und drücken <Leertaste> um in den Eingabemodus zu gelangen), wo Sie die Adresse bzw. den Adressbereich eines Sub-Netzwerks eingeben können (z.B. "197 . 200 . 100 . 255", wenn mit allen Knoten kommuniziert werden, die IP-Adressen im Bereich von "197 . 200 . 100 . x" haben).

Beachten Sie für Win32-Systeme, dass die Broadcast-Adressen mit der Subnet-Maske der TCP/IP-Konfiguration des Rechners übereinstimmen müssen !

Die folgenden Optionen können aktiviert oder deaktiviert werden, um das Übertragungsverhalten der Variablen zu konfigurieren:

Variablen packen: Die Variablen werden für die Übertragung in Pakete gebündelt (Telegramme), deren Größe vom Netzwerktyp abhängt. Wenn die Option deaktiviert ist, wird ein Paket pro Variable gebildet.

Prüfsumme übertragen: Jedem Variablenpaket wird eine Prüfsumme mitgegeben. Die Prüfsumme wird vom Empfänger überprüft, um sicherzustellen, dass die Variablendefinitionen von Sender und Empfänger übereinstimmen. Ein Paket mit nicht passender Prüfsumme wird nicht angenommen.

Zyklische Übertragung: Die Variablen werden innerhalb des angegebenen **Intervalls** gesendet. (Zeitangabe z.B. "T#70ms").

Übertragungon bei Änderung: Die Variablen werden nur gesendet, wenn sich ihre Werte geändert haben; mit dem **Mindestabstand** kann angegeben werden, wieviel Zeit mindestens zwischen zwei Übertragungen liegen muss..

Ereignisgesteuerte Übertragung: Die Variablen werden gesendet, sobald die angegebene **Variable** TRUE wird.

Beachten Sie, dass die Netzwerkvariablen automatisch bei jedem **Systemstart** versendet werden. Das bedeutet, dass die aktuellen Variablenwerte auch übertragen werden, wenn dies aufgrund der konfigurierten Übertragungs-Trigger (Änderung, Ereignis) zu diesem Zeitpunkt nicht geschehen würde.

Netzwerkeinstellungen

Wenn die Netzwerkfunktionalität vom Gerät unterstützt wird, können die aktuellen Netzwerk-Einstellungen für eine GNVL (Global Network Variable List) im Eigenschaften-Dialog angezeigt und verändert werden. Dabei handelt es sich um die Einstellungen, die beim Einfügen der GNVL im 'Objekt einfügen'-Dialog vorgenommen wurden.

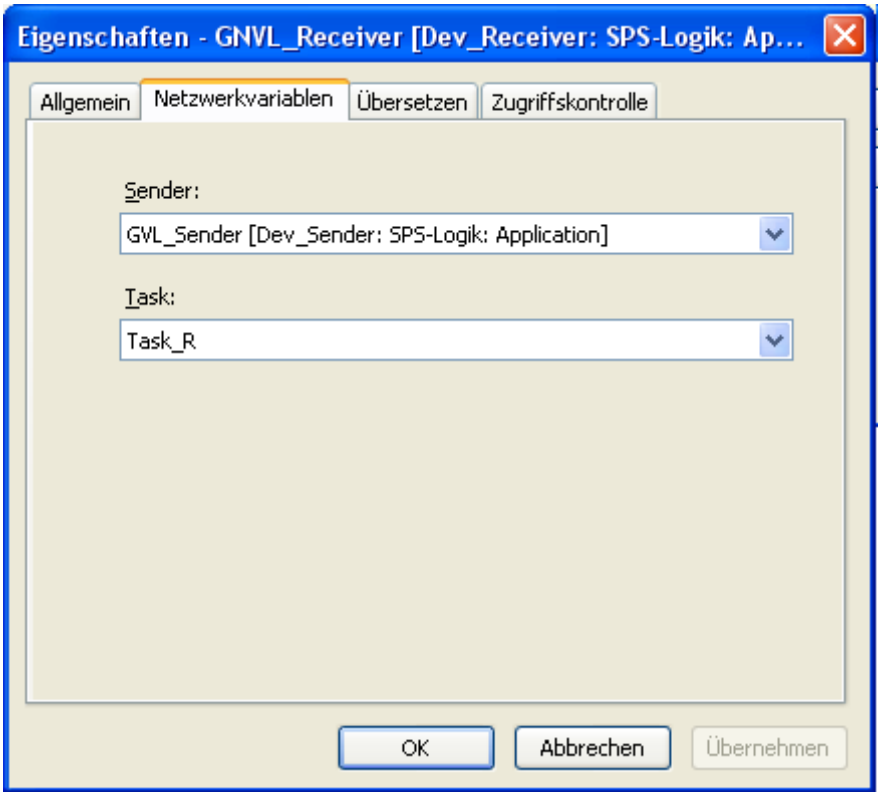


Abb. 292 Eigenschaften-Dialog, Kategorie Netzwerkeinstellungen

Sender: Name der Globalen Variablenliste des Sender-Geräts, die in der vorliegenden GNVL referenziert wird. Die eckigen Klammern enthalten den Namen des Geräts und der Applikation.

Task: Name der Task des aktuellen Geräts, die die Netzwerkvariablen kontrolliert.

AS-Einstellungen

Hier werden für das gerade ausgewählte AS-Objekt die aktuellen Einstellungen bezüglich des Übersetzens (Kompilieren) und der Handhabung der impliziten Variablen angezeigt und können verändert werden. Die auf den Registerblättern 'Variablen' und 'Übersetzen' behandelten Punkte entsprechen denen des 'AS'-Optionen-Dialogs, wo die Standardeinstellungen für AS-Objekte definiert sind.

Mit der Schaltfläche **Voreinstellungen** können die im AS-Optionen-Dialog definierten Standardwerte auf das gerade ausgewählte Objekt angewendet werden, d.h. sie werden hier im Eigenschaften-Dialog des Objektes dargestellt..

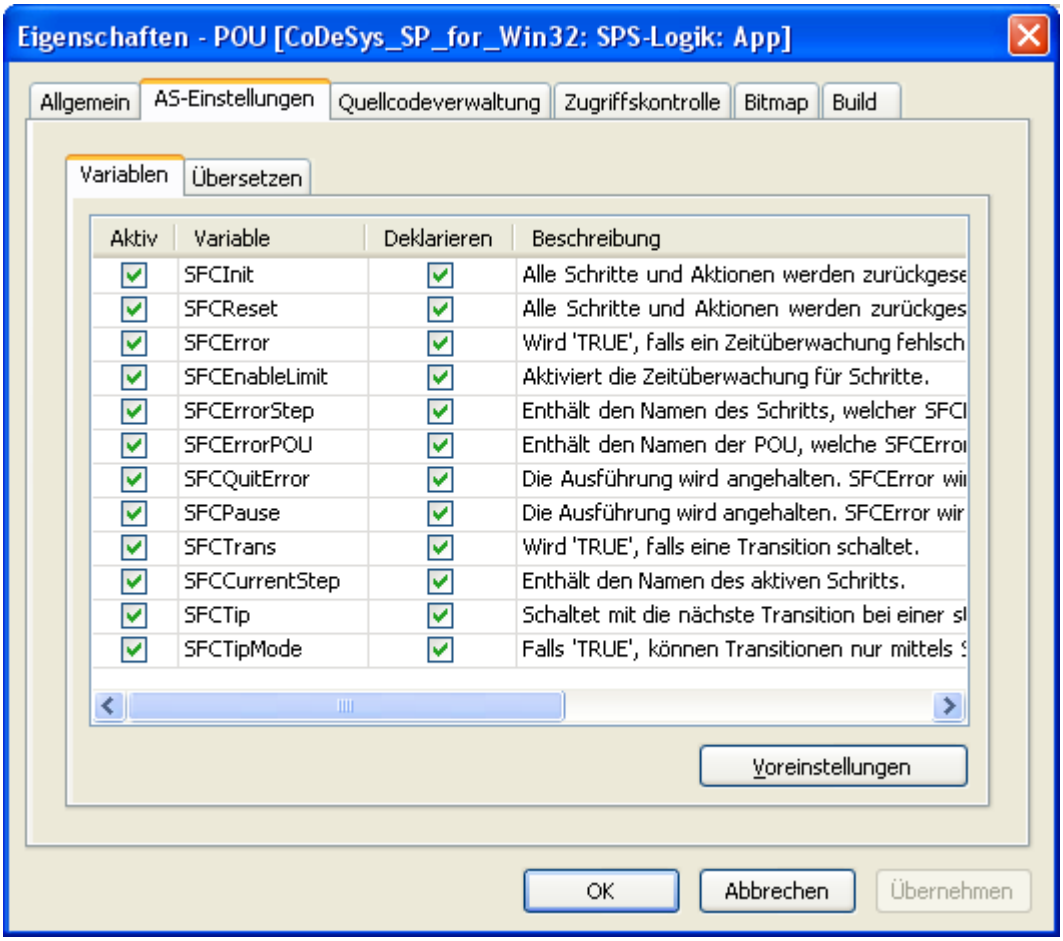


Abb. 293 Eigenschaften-Dialog, Kategorie AS-Einstellungen, Registerblatt "Variablen"

Verknüpfung mit Datei

Globale Variablenlisten können mit Hilfe einer externen Datei im Textformat definiert werden. Eine solche Datei kann über die Export.-Funktionalität erzeugt werden, die im Eigenschaften-Dialog der jeweiligen Variablenliste verfügbar ist. Wenn Option Export vor Übersetzen aktiviert ist, wird automatisch bei jeder Projektübersetzung (z.B. via F11) eine Datei mit Erweiterung "gvl" in dem Pfad gespeichert, der im Feld Dateiname angegeben ist. Wenn Option Import vor Übersetzen aktiviert ist, kann eine vorhandene Exportdatei bei jeder Projektübersetzung eingelesen werden. Dies ermöglicht, eine GVL, die aus einem anderen Projekt definiert wurde, zu importieren, um beispielsweise eine Kommunikation über Netzwerkvariablen aufzusetzen.

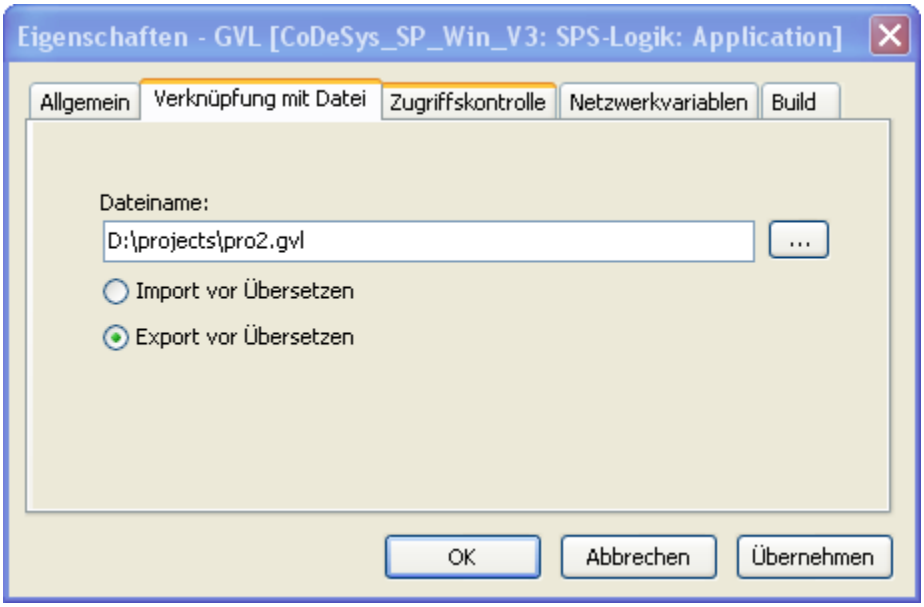


Abb. 294 Eigenschaften-Dialog, Kategorie Verknüpfung mit Datei

Sofern das aktuelle Profil die CAM Funktionalität enthält:

CAM

Hier werden die globalen Einstellungen der Kurvenscheibe vorgenommen, d.h. Dimension, Periode und Stetigkeitsanforderungen.

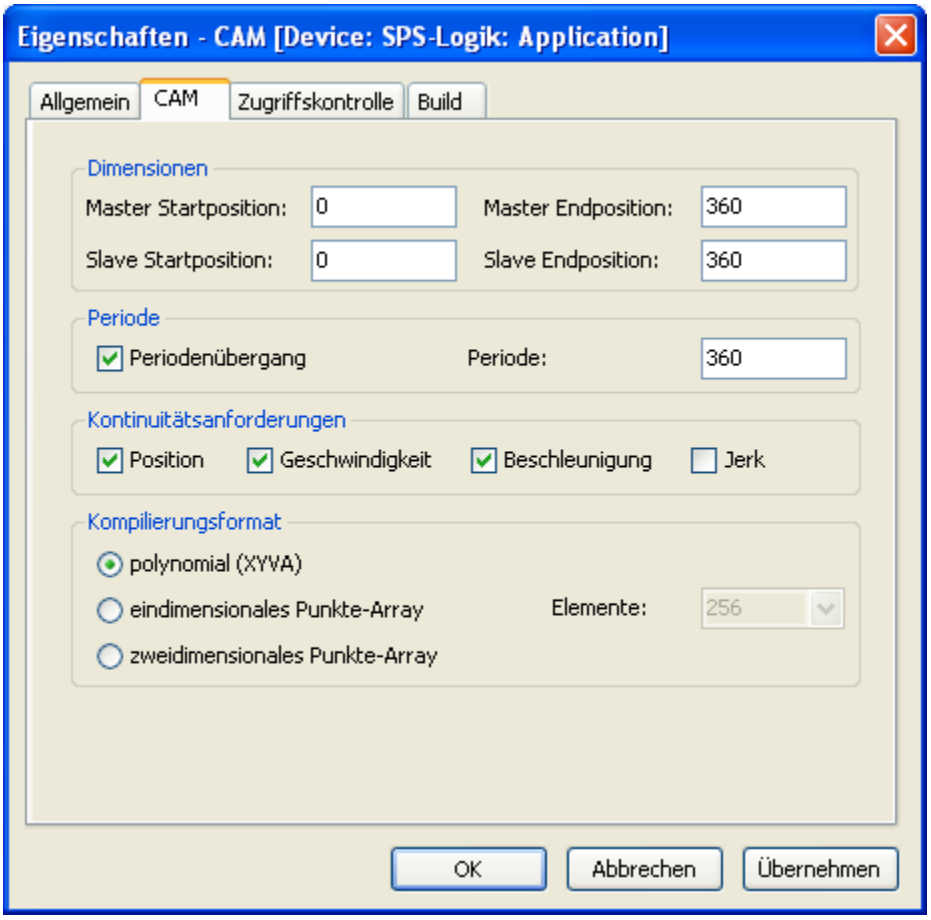


Abb. 295 Eigenschaftendialog für Kurvenscheibe

Dimensionen:

Master Startposition / Master Endposition: Start- und Endposition des Masters definieren den Umfang der Master-Werte und damit die Skala der horizontalen Achse der Kurvenscheibe. Die Standardeinstellungen erfolgen in Winkelgraden und somit werden 0 und 360 als Grenzwerte vorgegeben.

Slave Startposition / Slave Endposition: Die zugeordneten Slave-Positionen werden durch den Abbildungstyp, der für die Kurvenscheibe definiert ist, bestimmt. Jedoch kann der durch die Kurven dargestellte Ausschnitt, d.h. die Skala der vertikalen Achse, über die hier angegebenen Slave Start- und Endpositionen festgelegt werden.

Periode:

Periodenübergang: Wenn diese Option aktiviert ist , ist ein zyklisches Abbild der Kurvenscheibe vorgegeben. Das heißt, an der Endposition des Masters hat die zugehörige Slave-Position ein Vielfaches ihres Werts an der Startposition des Masters, während die Werte der Slave-Geschwindigkeit und Slave-Beschleunigung den gleichen Wert wie an der Startposition haben.

Periode: Hier kann die Slave-Periode ("Vorschub") in Einheiten der Slave-Achse festgelegt werden.

Kontinuitätsanforderungen:

Die Aktivierung dieser Optionen (**Position**, **Geschwindigkeit**, **Beschleunigung**, **Jerk**) bzgl. der Stetigkeit der Abbildung hat keine Auswirkung auf das Editieren der Kurve. Aber sie bewirkt eine Prüfung der Stetigkeit und eventuelle Verletzungen werden im Meldungsfenster (Kategorie "CAM" ausgegeben). Es ist nicht möglich, in der Positions-Abbildung Sprünge zu bearbeiten. Die Standardeinstellung fordert außerdem Kontinuität von Geschwindigkeit und Beschleunigung. Die Optionen können deaktivert werden (beispielsweise im speziellen Fall einer Abbildung, die nur aus linearen Segmenten besteht), was jedoch zu Knicken in der Positionskurve führen kann. Standardmäßig wird der Jerk, d.h. die vierte Ableitung nicht auf Sprünge getestet.

Kompilierungsformat:

polynomial (XYVA): polynomiale Beschreibung der einzelnen Punkte, bestehend aus Master-Position, Slave-Position, Slave-Geschwindigkeit und Slave-Beschleunigung

eindimensionales Punkte-Array: 1-dimensionale Tabelle von Slave-Positionen

zweidimensionales Punkte-Array: 2-dimensionale Tabelle von zusammengehörigen Master/Slave-Positionen

Ganzer Bildschirm

Symbol:

Standard Tastenkürzel: <Strg>+< Umsch>+< F12>

Wenn diese Option (Befehlskategorie 'Ansicht') aktiviert ist, wird das Hauptfenster der »PLC Designer« Benutzeroberfläche im Vollbildmodus angezeigt, d.h. bildschirmfüllend. Um zur vorher eingestellten Größe zurückzukehren, deaktivieren Sie die Option wieder oder drücken <Strg>+< Umsch>+< F12>.

Startseite

Symbol:

Dieser Befehl (Kategorie 'Ansicht') ist standardmäßig im Ansicht-Menü enthalten. Er öffnet eine Ansicht, die einige Befehle zum schnellen Starten mit einem neuen oder bestehenden Projekt sowie Versionsinformation und einen Viewer für die »PLC Designer« Homepage enthält.

Im Optionen-Dialog für Laden und Speichern kann konfiguriert werden, dass diese Startseite automatisch erscheint, nachdem das Programmiersystem gestartet wurde.

12.35.5 Meldungen

Befehle der Kategorie Ansicht - Meldungen

Das Plug-In 'Message View' stellt für Befehlskategorie 'Ansicht' den Befehl 'Meldungen' zum Öffnen des Meldungsfensters bereit. Standardmäßig ist der Befehl im Menü 'Ansicht' zu finden. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Dialogs 'Anpassen' umkonfiguriert werden.

Meldungen

Symbol:

Dieser Befehl (Kategorie Ansicht) öffnet das Meldungsfenster.

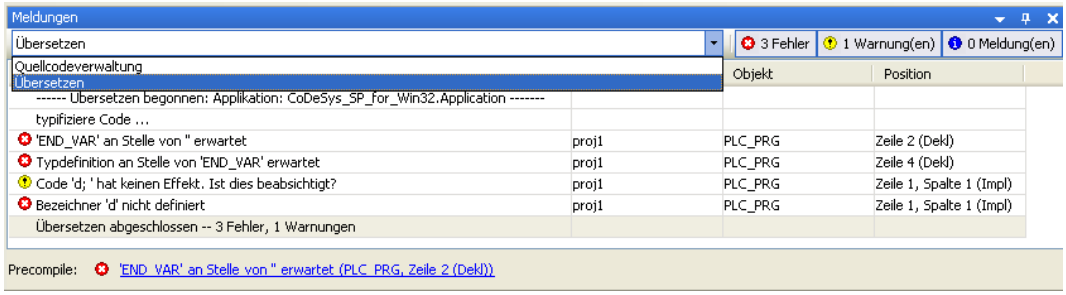


Abb. 296 Meldungsfenster, Beispiel

Meldungen können Fehler , Warnungen oder reine Information beschreiben.

Meldungen sind weiterhin nach Komponente oder Funktionalität kategorisiert. Beispielsweise gehören Meldungen bzgl. Syntaxprüfungen des Projekts in Kategorie 'Precompile', Meldungen zum Kompilieren des Projekts in Kategorie 'Übersetzen' (z.B. Übersetzungsfehler, Code-Größe). Auch Meldungen der Kategorien Import, Bibliotheksverwalter etc. sind möglich.

Ein Filter auf die gewünschte **Meldungskategorie** kann in der Auswahlliste oben im Meldungsfenster eingestellt werden (s.o. in der Abbildung), mit Ausnahme der **Precompile** Meldungen, die immer im Bereich unterhalb der eigentlichen Meldungstabelle angezeigt werden.

Die Meldungen der ausgewählten Kategorie werden in Tabellenform ausgegeben, mit folgenden Informationen: **Beschreibung** (Meldungstext), **Projekt** (Projektname), **Objekt** (Name des betroffenen Projektobjekts), **Position** (Stelle im Objekt, die die Meldung verursacht hat, z.B. Zeilennummer, Netzwerknummer etc.).

Wenn die Anzeige eines bestimmten Meldungstyps aus- oder eingeblendet werden soll, verwenden Sie die Schaltflächen in der oberen rechten Ecke des Fensters:: **Fehler**, **Warnungen**,(Informations)**Meldungen**. Die Schaltflächen zeigen auch immer die gerade verfügbare Anzahl von Meldungen des Typs an.

Zwischen den Meldungen im Meldungsfenster kann **hin- und hergesprungen** werden bzw. es kann von der gerade ausgewählten Meldung zur betreffenden **Position im Objekt** gesprungen werden. Dazu stehen die **Befehle** 'Nächte Meldung', 'Vorherige Meldung' und 'Gehe zu Quelltextposition' bereit. Durch einen Doppelklick auf einen Meldungseintrag in der Tabelle bzw. durch einen Mausklick auf die blau unterstrichene Precompile-Meldung gelangt man ebenfalls zur Quelltextposition.

12.35.6 AS Object

Befehle der Kategorie Ansicht - AS Objekt

Das Plug-In 'SFC Implementation Object' liefert den Befehl Elementeigenschaften (Befehlskategorie 'Ansicht'), mit dem der Eigenschaften-Dialog für das gerade ausgewählte AS-Element geöffnet wird.

Standardmäßig ist der Befehl im Menü 'Ansicht' enthalten. Wenn nötig, können sie den Anpassen-Dialog verwenden, um die Menükonfiguration zu verändern.

Elementeigenschaften

Symbol:

Dieser Befehl (Kategorie Ansicht) öffnet den **Eigenschaften**-Dialog für das gerade ausgewählte AS-Element. Die Elementeigenschaften, wie Name, Kommentar, Schrittzeiten und assoziierte Aktionen werden in einer gegliederten Tabelle dargestellt. Sie können in der Spalte 'Wert' editiert werden: Durch einen Mausklick auf das Feld gelangt man in den Eingabemodus bzw. kann man bei Schritten einen Haken im Kontrollkästchen setzen oder entfernen, um die Eigenschaft "Initialschritt" zu aktivieren oder zu deaktivieren.

12.35.7 Quellcodeverwaltung

Befehle der Kategorie Ansicht - Quellcodeverwaltung

Das Plug-In 'Source Control' stellt u.a. den Befehl 'Anstehende Eincheckvorgänge' für Befehlskategorie 'Ansicht' bereit. Dieser Befehl öffnet einen Dialog für das Handhaben der gerade aus der Projektdatenbank ausgecheckten Objekte.

Standardmäßig ist der Befehl im Menü 'Ansicht' zu finden. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs umkonfiguriert werden.

Anstehende Eincheckvorgänge

Dieser Befehl der Kategorie 'Ansicht' ist in Projekten, die unter Quellcodeverwaltung stehen, verfügbar. Standardmäßig ist er im Menü 'Ansicht' zu finden und öffnet den Dialog 'Anstehende Eincheckvorgänge'. Hier sind alle Projektobjekte aufgelistet werden, die gerade lokal ausgecheckt sind und einige Befehle für den Umgang mit diesen Objekten.

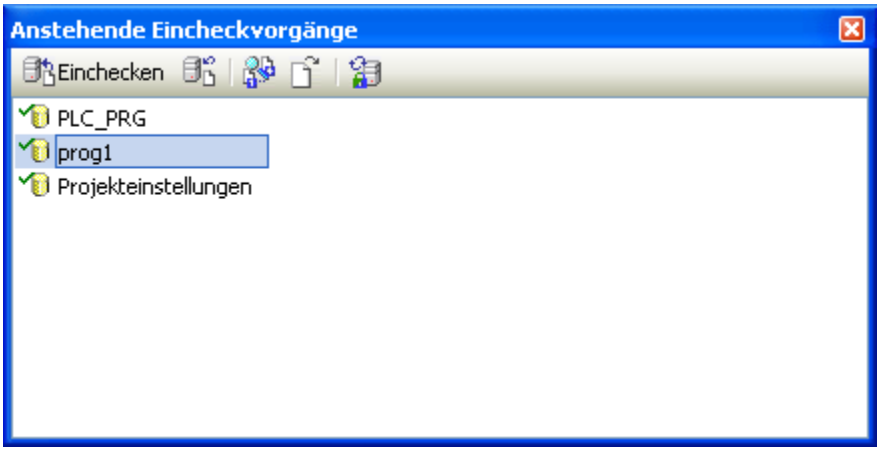


Abb. 297 Dialog 'Anstehende Eincheckvorgänge'

Die Werkzeugleiste im Dialog stellt Schaltflächen für die Befehle bereit, die auch im Menü Projekt / Quellcodeverwaltung zu finden sind. Mit Ausnahme des 'Aktualisieren'-Befehls wirken sie jeweils auf das gerade in der Liste ausgewählte Objekt. Wenn mehrere Objekte ausgewählt sind, sind nur die Befehle 'Einchecken' und 'Auschecken rückgängig' verfügbar.

Die Befehle:

Einchecken

Auschecken rückgängig

Unterschiede anzeigen

Editor öffnen

Status aktualisieren

12.35.8 Werkzeuge

Befehle der Kategorie Ansicht - Werkzeuge

Das Plug-In 'ToolBox View' stellt den Befehl Werkzeuge (Befehlskategorie 'Ansicht') bereit, der das Werkzeuge-Fenster ("Toolbox") für den gerade aktiven Editor öffnet.

Standardmäßig ist der Befehl im Menü 'Ansicht' zu finden. Wenn erforderlich kann die Menüstruktur mit Hilfe des Dialogs Anpassen umkonfiguriert werden.

Werkzeuge

Dieser Befehl (Kategorie Ansicht) öffnet ein Fenster mit "Werkzeugen" ("Toolbox") für den gerade aktiven Editor. Standardmäßig ist eine solche Werkzeugbox für grafische Editoren oder den Visualisierungs-Editor verfügbar. Sie enthält die grafischen Programmierelemente, die über "drag&drop" ins Editorfenster gezogen werden können.

Beispiele:

- Werkzeugbox für FUP/KOP/AWL-Editor
- Werkzeugbox für Visualisierungs-Edtior

12.35.9 Visualisierungselement

Befehle der Kategorie Ansicht - Visualisierungselement

Das Plug-In 'Visualization Element' stellt den Befehl Elementeigenschaften (Befehlskategorie 'Ansicht') bereit, der für das gerade in der Visualisierung selektierte Element das Fenster 'Eigenschaften' öffnet. Dies dient der Ansicht und Bearbeitung der jeweiligen Elementeigenschaften.

Standardmäßig ist der Befehl im Ansicht-Menü zu finden. Wenn erforderlich, kann mit Hilfe des Anpassen-Dialogs die Menüstruktur verändert werden.

Elementeigenschaften

Symbol:

Dieser Befehl (Kategorie Ansicht) öffnet das **Eigenschaften**-Fenster (Ansicht) für das gerade selektierte Visualisierungselement. Die Elementeigenschaften werden in Gruppen sortiert in einer Tabelle dargestellt. Durch Mausklicks auf die Plus- bzw. MInuszeichen vor dem Gruppennamen können die zugehörigen Parameter angezeigt oder ausgeblendet werden.

Es hängt vom Visualisierungselement ab, welche Eigenschaften konfiguriert werden können.

Hinweis: Der Befehl kann auch in anderen grafischen Editoren verfügbar sein, wo er ebenfalls die Eigenschaften-Ansicht für ein gerade selektiertes grafisches Element öffnet (z.B. AS-Editor).

12.35.10 Überwachen

Befehle der Kategorie Ansicht - Überwachen

Das Plug-In 'Watchlist View' stellt den Befehl 'Überwachen' (Befehlskategorie 'Ansicht') zum Öffnen einer Überwachungsliste bereit.

Standardmäßig ist der Befehl im Ansicht-Menü zu finden. Wenn erforderlich, kann die Menüstruktur mit Hilfe des Anpassen-Dialogs verändert werden.

Überwachen

Dieser Befehl (Kategorie Ansicht) öffnet ein Untermenü mit den Befehlen **Überwachen 1**, **Überwachen 2**, **Überwachen 3** und **Überwachen 4** oder **Alle Forces betrachten**. Damit kann die jeweilige Überwachungsliste in einem Fenster geöffnet und bearbeitet werden. 'Alle Forces betrachten' ist eine spezielle Ansicht aller aktuell geforcten Werte.

Standardmäßig ist dieses Kommando Teil des Menüpunkts 'Ansicht'. Falls nötig, öffnen Sie den Dialog Anpassen... um die Menükonfiguration einzusehen oder zu verändern (sh. Kategorie 'Ansicht').

12.36 Fenster

Die Befehlskategorie 'Fenster' enthält Befehle zur Handhabung der in der Benutzeroberfläche verwendeten Fenster (Öffnen, Anordnen etc.).

12.36.1 Fenster Befehle - »PLC Designer« Frame

Der "Frame" stellt Befehle für Kategorie 'Fenster' bereit, mit denen die Fenster der Benutzeroberfläche gehandhabt werden können. Die Befehle gelten nur für tatsächliche "Fenster", die - im Gegensatz zu "Ansichten (Views)" beispielsweise eines Editors - in ihrer Menüleiste am oberen Rand den "Docking Button" enthalten.

Standardmäßig sind die Befehle im Menü 'Fenster' zu finden. Falls nötig, kann die Menüstruktur über den Anpassen-Dialog umkonfiguriert werden.

Die Befehle:

- Andocken
- Unverankert
- Ausblenden
- Nächster Editor
- Vorheriger Editor
- Nächstes Teilfenster
- Vorheriges Teilfenster
- Editor schließen
- Alle Editoren schließen
- Fenster-Layout zurücksetzen
- Neue horizontale Registerkartengruppe
- Neue vertikale Registerkartengruppe
- Fenster <n>

12.36.2 Andocken

Dieser Befehl (Kategorie 'Fenster') dient dazu, ein Fenster, das vorher mit dem Befehl Unverankert gelöst wurde und gerade als unverankertes Fenster auf dem Bildschirm liegt, wieder an den Rahmen der Benutzeroberfläche "anzudocken", d.h. zu fixieren.

12.36.3 Unverankert

Dieser Befehl (Kategorie 'Fenster') dient dazu, ein Fenster, das am Rahmen der Benutzeroberfläche angedockt (fixiert) ist, vom Rahmen zu lösen und als unverankertes Fenster auf dem Bildschirm zu platzieren, was dann auch außerhalb der Benutzeroberfläche möglich ist. Um ein unverankertes Fenster wieder an den Rahmen der Benutzeroberfläche zu binden, verwendet man den Befehl Andocken.

12.36.4 Ausblenden

Dieser Befehl (Kategorie 'Fenster') dient dazu, ein Fenster "aus- oder einzublenden". Ausblenden bedeutet, dass das Fenster nur mehr als ein Reiter im Rahmen der Benutzeroberfläche vertreten ist und nur angezeigt wird, wenn mit der Maus auf diesen Reiter geklickt wird. Der Befehl funktioniert als Option, d.h. bei einem ausgeblendeten Fenster erscheint er mit einem davor gesetzten Haken im Menü. Ein erneutes Anklicken entfernt den Haken und blendet das Fenster wieder ein.

Der Befehl entspricht der Verwendung des **"docking buttons"** in der oberen rechten Ecke eines Fensters.

12.36.5 Nächster Editor

Standard Tastaturkürzel: <Strg> + <F6>

Wenn mehrere Editor"fenster" (Ansicht) geöffnet sind, kann dieser Befehl (Kategorie 'Fenster') verwendet werden, um den Fokus vom gerade aktiven zum nächsten Fenster zu verschieben. Das "nächste" Fenster ist dabei das, dessen Reiter rechts vom gerade aktiven in der Registerkartengruppe liegt.

12.36.6 Vorheriger Editor

Standard Tastaturkürzel: <Umsch> + <Strg> + <F6>

Wenn mehrere Editor"fenster" (Ansicht) geöffnet sind, kann dieser Befehl (Kategorie 'Fenster') verwendet werden, um den Fokus vom gerade aktiven zum vorherigen Fenster zu verschieben. Das "vorherige" Fenster ist dabei das, dessen Reiter links vom gerade aktiven in der Registerkartengruppe liegt.

12.36.7 Nächstes Teilfenster

Standard Tastaturkürzel: <F6>

Dieser Befehl (Kategorie 'Fenster') kann in "Fenstern" verwendet werden, die zwei oder mehr Teilfenster enthalten, um den Fokus ins nächste Teilfenster zu setzen.

Beispiel: Wenn ein Objekt im ST-Editor geöffnet ist und der Cursor gerade im Deklarationsfenster steht, wird der Fokus mit Befehl 'Nächstes Teilfenster' in den Implementationsteil gesetzt.

Siehe auch: Vorheriges Teilfenster

12.36.8 Vorheriges Teilfenster

Standard Tastaturkürzel: <Strg> + <F6>

Dieser Befehl (Kategorie 'Fenster') kann in "Fenstern" verwendet werden, die zwei oder mehr Teilfenster enthalten, um den Fokus ins vorherige Teilfenster zu setzen.

Beispiel: Wenn ein Objekt im ST-Editor geöffnet ist und der Cursor gerade im Implementationsteil steht, wird der Fokus mit Befehl 'Vorheriges Teilfenster' ins Deklarationsfenster gesetzt.

Siehe auch: Nächstes Teilfenster

12.36.9 Editor schließen

Standard Tastaturkürzel: <Strg> + <F4>

Dieser Befehl (Kategorie 'Fenster') schließt das gerade aktive Editorfenster.

12.36.10 Alle Editoren schließen

Symbol:

Dieser Befehl (Kategorie 'Fenster') schließt alle gerade geöffneten Editorfenster.

12.36.11 Fenster-Layout zurücksetzen

Dieser Befehl (Kategorie 'Fenster') setzt alle gerade geöffneten Fenster und Ansichten auf ihre Standardpositionen zurück. Eine Bestätigung wird vom Anwender verlangt, bevor dies erfolgt.

Siehe hierzu auch die Beschreibung über das Anpassen der Benutzeroberfläche für generelle Informationen zum Anordnen von Fenstern und Ansichten.

12.36.12 Neue horizontale Registerkartengruppe

Symbol:

Dieser Befehl (Kategorie 'Fenster') ist verfügbar, wenn mehrere Editorfenster als Registerblätter nebeneinander angeordnet sind.

Er veranlaßt, dass das gerade aktive Fenster in eine neue, separate Registerkartengruppe unterhalb der bereits existierenden, verschoben wird.

Wenn ein weiteres Objekt zur Bearbeitung geöffnet wird, d.h. ein weiteres Editorfenster aufgemacht wird, wird dies automatisch in die Registerkartengruppe eingeordnet, in der zuletzt der Fokus lag.

Siehe auch: Neue vertikale Registerkartengruppe

12.36.13 Neue vertikale Registerkartengruppe

Symbol:

Dieser Befehl (Kategorie 'Fenster') ist verfügbar, wenn mehrere Editorfenster als Registerblätter nebeneinander angeordnet sind.

Er veranlaßt, dass das gerade aktive Fenster in eine neue, separate Registerkartengruppe rechts neben der bereits existierenden, verschoben wird.

Wenn ein weiteres Objekt zur Bearbeitung geöffnet wird, d.h. ein weiteres Editorfenster aufgemacht wird, wird dies automatisch in die Registerkartengruppe eingeordnet, in der zuletzt der Fokus lag.

Siehe auch: Neue horizontale Registerkartengruppe

12.36.14 Fenster <n>

Für jedes geöffnete Editorfenster enthält das Menü 'Fenster' einen Befehl '<Objektname>' (Kategorie 'Fenster'), über den das Fenster aktiviert, also der Fokus dorthin gesetzt werden kann. Bei "offline"-Ansichten wird hinter dem Befehl "(Offline)" angezeigt, im Fall von Funktionsblock-Ansichten wird "(Impl)" bzw. "<Instanzpfad>" zur Unterscheidung ergänzt.

12.36.15 Fenster...

Dieser Befehl (Kategorie 'Fenster') öffnet den Dialog 'Fenster', in dem alle gerade geöffneten Editorfenster aufgelistet werden.

Um eines der Fenster zu aktivieren, also den Fokus dorthin zu setzen, wählen Sie den entsprechenden Eintrag aus und betätigen Schaltfläche **Aktivieren**.

Um ein oder mehrere Fenster zu schließen, wählen Sie die Einträge aus und betätigen Schaltfläche **Fenster schließen**. Mehrfachauswahl ist möglich.

Horizontal anordnen: Derzeit nicht verfügbar.

Vertikal anordnen: Derzeit nicht verfügbar.

Schließen schließt den Dialog.

13 Allgemeine Editoren

13.1 CAM Editor

Der CAM Editor steht nur für »PLC Designer« SoftMotion Steuerungen zur Verfügung. Diese setzen die Installation des zusätzlichen Softwarepakets »PLC Designer« SoftMotion voraus.

13.1.1 Eine kurze Übersicht

Die SoftMotion Kurvenscheibe ist in die Programmieroberfläche von »PLC Designer« integriert. Innerhalb des CAM Editors können **Kurvenscheiben** (CAMs) und Nocken jeweils sowohl graphisch als auch mit Hilfe von Tabellen implementiert werden. Sobald für die zugehörige Applikation Code erzeugt wird, werden globale Datenstrukturen (CAM Data) angelegt, auf die das IEC Programm zugreifen kann. Zu diesem Zweck wird automatisch mit dem Einfügen eines SoftMotion-Antriebs auch die Bibliothek SM3_Basic.library in das Projekt eingebunden.

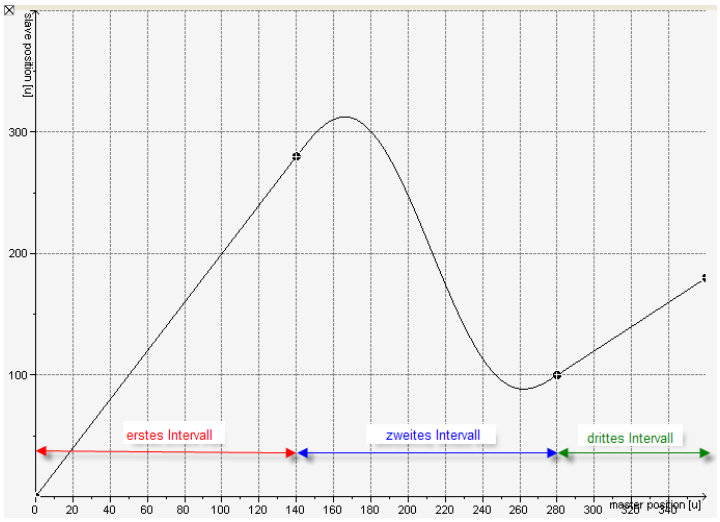
(Hinweis: Beachten Sie die Beispielprogramme und Allgemeinen Hinweise zu Kurvenscheiben)

13.1.2 Definition einer SoftMotion Kurvenscheibe

Eine Kurvenscheibe beschreibt die funktionale Abhängigkeit eines Parameters (Slave) von einem anderen (Master). Die Beziehung wird beschrieben durch eine stetige Funktion (oder Abbildung), welche einen definierten Bereich an Master-Werten auf Slave-Werte abbildet. Noch genauer: Nach Unterteilen der Master-Achse in geeignete Abschnitte kann der Graph dieser Funktion auf jedem dieser Teilintervalle entweder durch eine Gerade oder aber durch ein Polynom 5. Grades dargestellt werden.

Beispiel eines Kurvenscheiben-Graphs

Graph einer Kurvenscheibe:die Master-Werte sind auf der horizontalen, die Slave-Werte auf der vertikalen Achse aufgetragen.



Im Beispiel liegen die Master-Werte zwischen 0 und 360. Dieser Bereich ist in drei Intervalle unterteilt:

- das erste auf der linken Seite: [0, 140]
- das zweite in der Mitte: [140, 280]
- das dritte auf der rechten Seite: [280, 360]

Auf dem ersten und dritten Intervall ist die Funktion (Abbildung) linear, ihr Graph entsprechend ein Geradenstück. Demzufolge ist ihre erste Ableitung (d.i. die Steigung) konstant, alle höheren Ableitungen sind 0.

Auf dem mittleren Intervall wird die Abbildung durch ein Polynom fünften Grades beschrieben. Somit ist ihre erste Ableitung ein Polynom vierten Grades, ihre zweite Ableitung (d.i. die Krümmung) ein Polynom dritten Grades, ihre dritte Ableitung ein Polynom zweiten Grades (usw.).

Wenn die Funktion die Bewegung des Slaves in Abhängigkeit von der Position des Masters beschreibt, so entspricht ihre erste Ableitung gerade der Geschwindigkeit des Slaves, die zweite Ableitung seiner Beschleunigung.

Behält man diese physikalische Interpretation im Hinterkopf, so wird offensichtlich, dass die Abbildung stetig sein muss, was bedeutet, dass ihr Graph keine Sprünge aufweist. Insbesondere muss die Stetigkeit auch an jedem Punkt erfüllt sein, an welchem zwei Teilintervalle zusammentreffen. Des Weiteren wird die Stetigkeit i.a. auch von der ersten und zweiten Ableitung gefordert. (In der Tat bestimmen diese drei Stetigkeitsforderungen an Anfangs- und Endpunkt eines Intervalls gerade die Koeffizienten des zwischen zwei Geradenstücken eingefügetn Polynom fünften Grades.)

Ist die Kurvenscheibe als 'periodic transition' (periodische Transition) ausgelegt, so müssen zusätzlich die Werte von Funktion und Ableitungen für Start- und Endpunkt des Master-Wertebereichs übereinstimmen (wobei der Wert der Funktion am Endpunkt auch ein Vielfaches des Wertes am Startpunkt sein darf).

Außerdem dürfen an beliebigen Stellen Nocken, i.e. binäre Schalter, zur Kurvenscheibe hinzugefügt werden. Insbesondere können **Nockenschaltwerke**, also Kurvenscheiben-Tabellen, die nur Nocken enthalten, angelegt werden, wobei die Slave-Position über den gesamten Master-Wertebereich auf 0 gesetzt wird.

13.1.3 Anlegen einer Kurvenscheibe

Eine Kurvenscheibe kann als Objekt in ein Projekt oder eine Applikation eingebunden werden. Selektieren Sie hierzu das Projekt oder die Applikation (im Geräte- oder POUs-Fenster) und wählen Sie den Befehl **Objekt hinzufügen...** aus dem Kontextmenü, woraufhin der Dialog "Objekt hinzufügent" geöffnet wird.

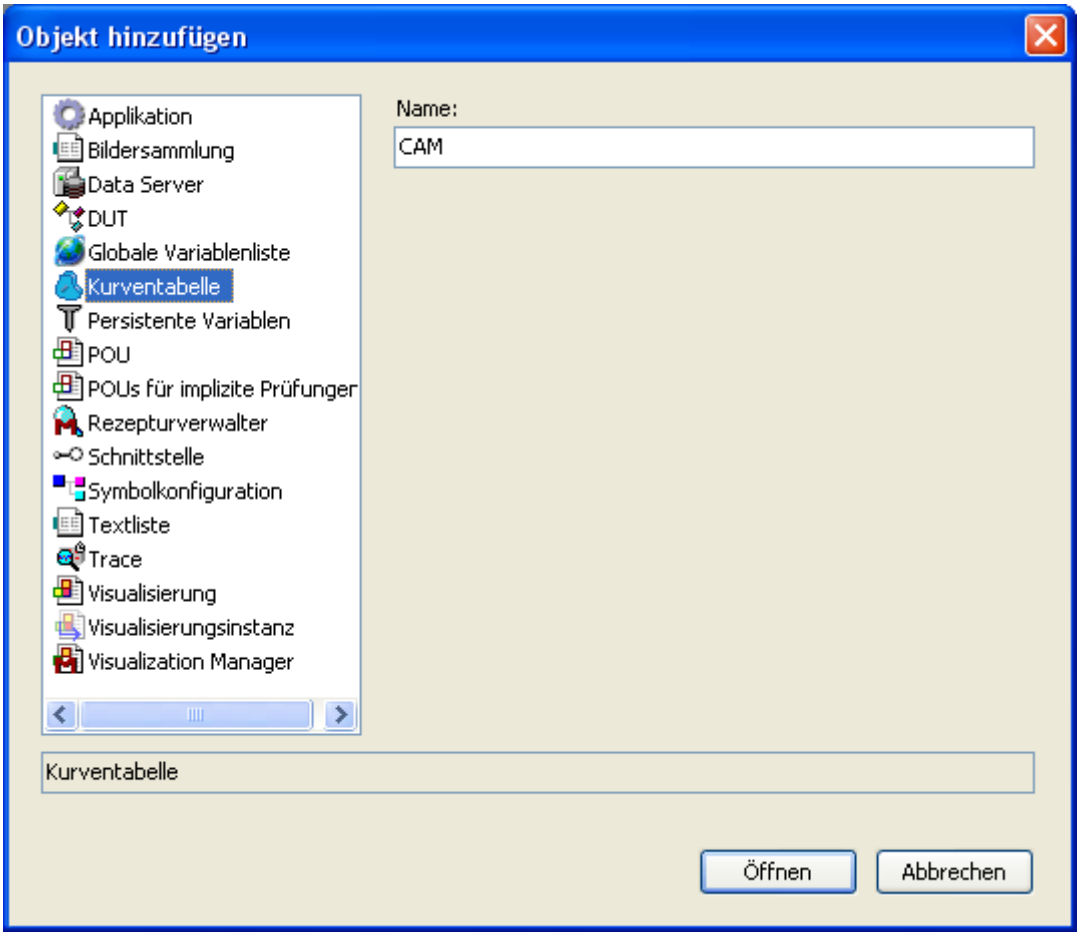


Abb. 298 Dialog 'Objekt hinzufügen'

Wählen Sie hier das Objekt "CAM table", geben Sie einen Namen für die Kurvenscheibe ein (standardmäßig "CAM") und bestätigen Sie Ihre Auswahl mit **Öffnen**. Daraufhin wird die Kurvenscheibe in den Gerätebaum (oder bei den POUs) eingefügt, der Kurvenscheiben-Editor öffnet und zeigt die Standardeinstellungen für eine neu angelegte Kurvenscheibe an.

13.1.4 Editieren einer Kurvenscheibe

Der Kurvenscheiben-Editor besteht aus vier Teilen, wovon die ersten beiden automatisch beim Anlegen einer Kurvenscheibe geöffnet werden:

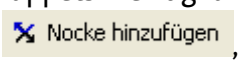
- eine **Toolbox** mit **Cam Editor Werkzeugen** zur Selektion und dem Einfügen von Punkten. Siehe unten, Werkzeugbox.
- ein **Haupteditor** überschrieben mit dem Namen der Kurvenscheibe gefolgt von ihrem Pfad im Gerätebaum (z. B. CAM [PLCWinNT:SPS-Logik:Applikation]). Dieser Editor enthält vier Registerkarten, welche die Kurvenscheiben entweder graphisch oder mit Hilfe von Tabellen aufzeigen. Siehe unten, Arbeiten im Haupteditor.
- ein Dialog 'Elementeigenschaften', der mittels des Eintrags **Elementeigenschaften** geöffnet wird (standardmäßig unter dem Menüpunkt **Ansicht**). Siehe unten, Editieren im Elementeigenschaften-Editor.
- ein Dialog 'Eigenschaften', welcher allgemeine Informationen über das Objekt Kurvenscheibe enthält. Er wird über den Eintrag **Eigenschaften...** im Kontextmenü der Kurvenscheibe geöffnet oder mittels des Befehls **Eigenschaften...** (standardmäßig unter dem Menüpunkt **Ansicht**). Siehe unten, Editieren im Kurvenscheiben-Eigenschaften-Editor.

Die Werkeugbox (Toolbox)

Abhängig von der aktuell angezeigten Registerkarte des Haupteditors enthält die Toolbox folgende Kurvenscheiben-Werkzeuge:

- das **Auswählen Werkzeug** steht immer zur Verfügung. Aktivieren Sie es durch Anklicken von

, woraufhin ein Mausklick innerhalb des Haupteditors in den graphischen Darstellungen Punkte, Nocken oder Funktionsstücke auswählt, in den tabellarischen Darstellungen entsprechend Einträge. Selektierte Punkte oder Funktionsstücke werden innerhalb der graphischen Darstellung fett dargestellt, wohingegen selektierte Tabelleneinträge durch eine dünnen blauen Rahmen kenntlich gemacht werden (zusätzlich zu einer blauen Einfärbung der zugehörigen Tabellenzeile).
- das **Punkt hinzufügen** Werkzeug steht nur in der Registerkarte mit dem Titel "CAM" zur Verfügung. Aktivieren Sie dieses W7erkzeug durch einen Mausklick auf

, woraufhin dann bei einem Mausklick im Haupteditor ein neuer Punkt im Positions-, Geschwindigkeits- und Beschleunigungsgraphen eingefügt wird. Beachten Sie, dass der Jerk-Graph nicht editierbar ist.
- das **Nocke hinzufügen** Werkzeug ist nur im Registerblatt "Tappets" verfügbar. Aktivieren Sie dieses Werkzeug durch einen Mausklick auf 

, woraufhin dann bei einem Mausklick im Haupteditor eine neue Nocke im jeweiligen Track eingefügt wird..

Sehen Sie in der folgenden Beschreibung zum Haupteditor detailliertere Informationen zur Anwendung der Werkzeuge.

Arbeiten im Haupteditor

Der Haupteditor enthält vier Registerblätter, auf denen der Kurvenscheiben-Graph und seine Ableitungen bzw. die hinzugefügten Nocken entweder graphisch oder tabellarisch dargestellt und bearbeitet werden. Änderungen im graphischen Editor werden automatisch in den tabellarischen Editor übernommen, und umgekehrt. Der Bearbeitungsmodus hängt vom jeweiligen Registerblatt ab:

Registerblatt "CAM": Graphischer Editor für die Definition des Kurvenscheiben-Graphs. Das Editorfenster ist in vier Unterfenster unterteilt, in denen (von oben nach unten) die Slave-Position in schwarz, die Geschwindigkeit in Blau, die Beschleunigung in Grün und der Jerk in Gelb dargestellt wird.

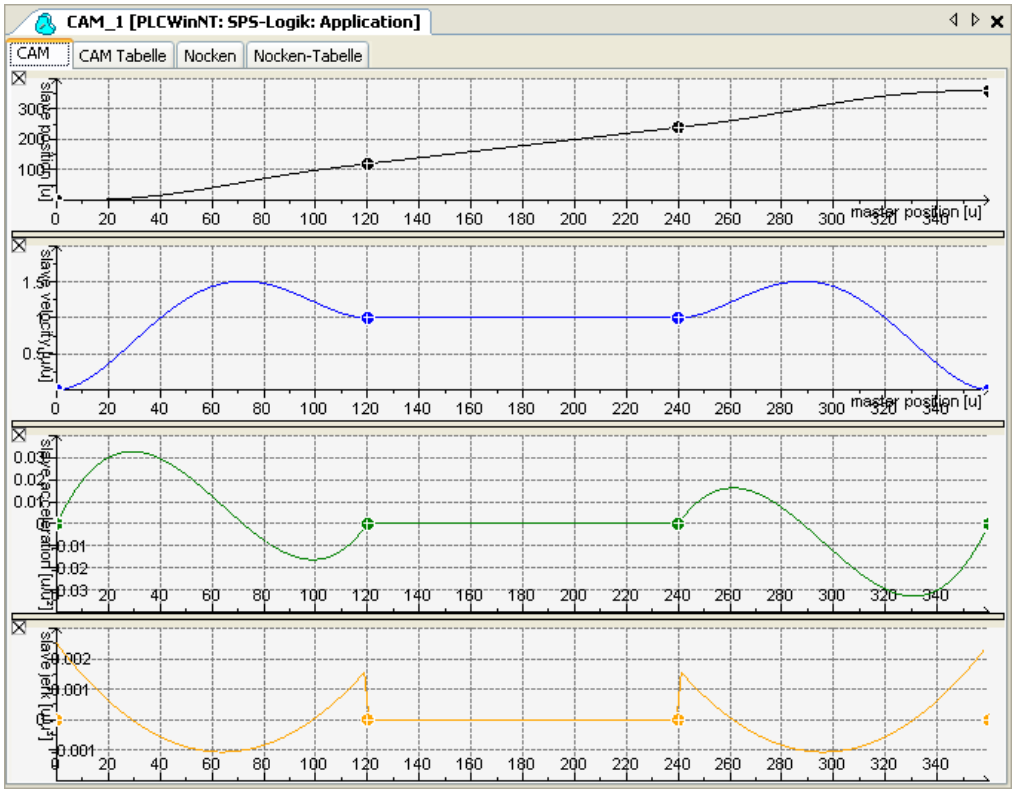


Abb. 299 Haupteditor: CAM

Die Höhe der Unterfenster kann durch Verschieben der horizontalen Trennbalken mit der Maus verändert werden. Ein Unterfenster kann durch einen Mausklick auf das ☒ Symbol in der oberen linken Ecke auch geschlossen werden. Anstelle des Fensters wird dann nur der Titel ("Position", "Geschwindigkeit", "Beschleunigung" bzw. "Jerk") angezeigt. Durch einen Mausklick auf diesen Text kann das Unterfenster wieder geöffnet werden.

In jedem Unterfenster zeigt die horizontale Achse des Koordinatensystems standardmäßig den Bereich der Master-Werte ([0,360]. Im Positions-Diagramm ist die vertikale Achse gemäß den Einstellungen in den CAM-Objekteigenschaften skaliert, während sie in den anderen Diagrammen automatisch entsprechend des jeweiligen Wertebereichs von Geschwindigkeit, Beschleunigung bzw. Jerk. skaliert ist.

Mit Ausnahme der Jerk-Kurve sind alle Diagramme editierbar. Da Geschwindigkeit, Beschleunigung und Jerk abgeleitete Kurven sind, wirkt sich eine Veränderung in einem der Graphen jeweils auch auf die anderen aus.

Die **Standard-Abbildung** einer neu eingefügten Kurvenscheibe enthält **vier Punkte**, die **drei Abschnitte** [0,120], [120,240] und [240,360] definieren. Die zugeordneten Slave-Positionen entsprechen den Master-Positionen, die zugehörigen Geschwindigkeits-, Beschleunigungs- und Jerk-Werte sind Null außer für den zweiten und dritten Punkt, denen eine Geschwindigkeit von 1 zugewiesen ist. Jedes der Intervall-Stücke des Kurvenscheiben-Graphen ist vom Typ **Poly5** (d.h. ein Polynom fünften Grades).

Diese Standard-Abbildung kann folgendermaßen verändert werden:

Um einen weiteren Punkt einzufügen wählen Sie **Punkt hinzufügen** aus der Werkzeugbox und klicken auf die gewünschte x-Position (muss nicht direkt auf der x-Achse erfolgen) in einem der oberen drei Diagramme. Der Abschnitt, der den Master-Wert des neuen Punktes enthält, wird in zwei Abschnitte aufgeteilt und die Anfangs- und Endwerte der neuen Abschnitte (vom Typ Poly5) werden entsprechend angepaßt.

Um die einem Punkt zugewiesene Position, Geschwindigkeit oder Beschleunigung zu modifizieren, wählen Sie **Auswählen** aus der Werkzeugbox, klicken auf den Punkt in einem der oberen drei Diagramme und verschieben ihn bei gedrückter Maustaste. Alternativ können Sie den elementspezifischen Eigenschaften-Editor verwenden.

Um einen Punkt zu löschen, wählen Sie ihn mit einem Mausklick aus und drücken die <Entf>-Taste oder verwenden den Befehl Entfernen.

Registerblatt "CAM Tabelle": Tabellarischer Editor für die Definition des Kurvenscheiben-Graphs. Die ersten fünf Spalten der hier gezeigten Tabelle enthalten Attribute der Punkte (d.h. **X**- und **Y**-Wert der Position, die zugehörige Geschwindigkeit **V**, die Beschleunigung **A** und Jerk **J**). Die nächsten fünf Attribute sind die der Abbildungsabschnitte (d.h. **Segment type**, die **Minimal**- und **Maximal**werte der Slave-Position und der Maximalwert von Geschwindigkeit und Beschleunigung innerhalb des betreffenden Abschnitts). Die erste Zeile der Tabelle enthält immer die **Startposition des Masters** (und die darauf bezogenen Slave-Werte), die letzte immer seine Endposition. Die immer ungerade Anzahl der dazwischen liegenden Zeilen (mindestens eine!) definieren abwechselnd Abschnitte und/oder Punkte.

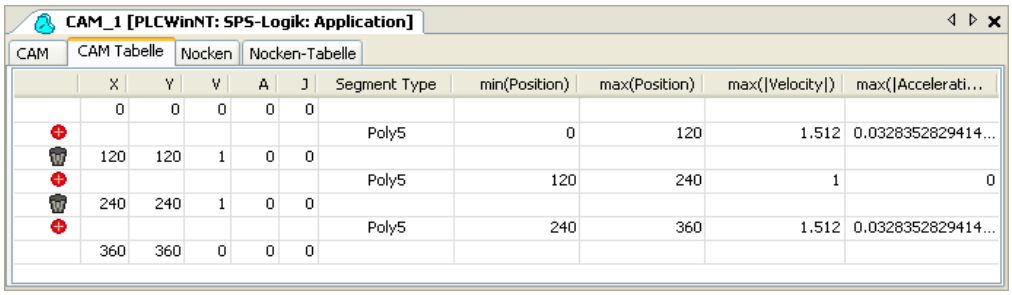


Abb. 300 Haupteditor: CAM Tabelle

Um in einem bestehenden Abschnitt (Segment) einen **weiteren Punkt einzufügen**, klicken Sie auf das Symbol in der Zeile, die der mit den Attributen des betreffenden Abschnitts vorangeht. Daraufhin werden zwei neue Zeilen unterhalb eingefügt: die erste mit Einträgen zum Mittelpunkt des bestehenden Abschnitts, die zweite mit den Attributen des neu hinzukommenden Abschnitts. Unabhängig davon, welchen Typ das Segment vor dem Einfügen des neuen Punktes hatte, wird der **Segmenttyp** jetzt für die beiden neu entstehenden Abschnitte automatisch auf Poly5 gesetzt.

Um **einen Tabelleneintrag zu verändern** wählen Sie die entsprechende Zelle durch einen Mausklick aus. Beachten Sie, dass die Maximal-Werte in den letzen vier Spalten aus den Einstellungen im jeweiligen Abschnitt resultieren und somit nicht editiert werden können. In einer ausgewählten Zelle kann durch die <Eingabe>-Taste ein Editierfeld bzw. eine Auswahlliste geöffnet werden und der gewünschte Wert durch Eintippen bzw. Auswählen eingegeben werden. Achten Sie dabei auf die Einhaltung der Kontinuität der Kurve! Wenn diese durch die Eingabe verletzt wird, wird die Eingabe entweder nicht angenommen oder es ergeben sich automatische Veränderungen anderer Tabelleneinträge.

Um **einen Punkt zu löschen** (solange es sich nicht um die Start- oder Endposition des Masters handelt), klicken Sie auf das Symbol in der vorangehenden Tabellenzeile. Diese und die Punkt-beschreibende Zeile werden gelöscht. Der Abschnitt links des gelöschten Punktes wird mit dem rechts davon verschmolzen. Das resultierende Segment ist vom Typ Poly5, unabhängig davon, welchen Typ die beiden Abschnitte vorher hatten.

Registerblatt "Nocken": Graphischer Editor für die Definition von Nockenspuren. Eine Nockenspur definiert eine oder mehrere Master-Postions-abhängige Nocken, d.h. Nocken-Ereignisse, für denselben Nockenschalter. Am oberen Rand des Editorfensters stellt eine horizontale Achse den Bereich der Master-Positionen an. Unterhalb folgen die einzelnen Nockenspuren, vorangestellt jeweils die "Track-ID"-Nummer, die über die Pfeiltasten nach oben oder unten verändert werden kann (standardmäßig sind zwei Tracks vorgegeben, der erste davon bereits mit zwei Nocken ausgestattet).

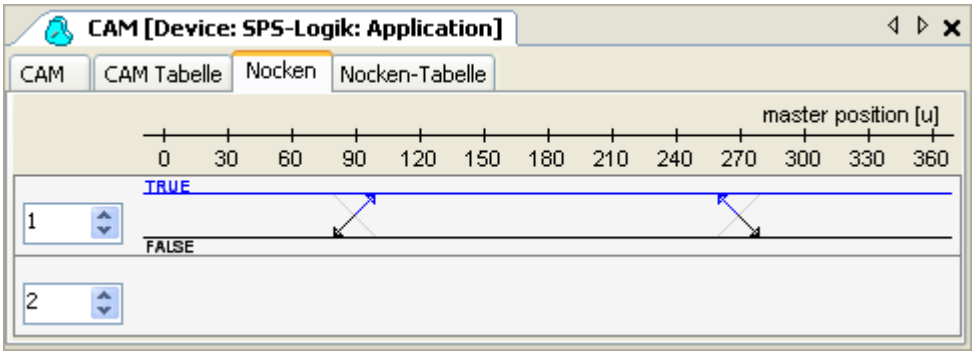


Abb. 301 Hautpeditor: Nocken

Wählen Sie **Nocke hinzufügen** aus der Werkzeugbox, um **eine neue Nocke hinzuzufügen**. Danach klicken Sie in der gewünschten Spur auf die gewünschte Einfügeposition. Die neue Nocke wird eingefügt und automatisch **Auswählen** in der Werkzeugbox aktiviert. Die Nocke kann nun mit einem Mausklick selektiert und bei weiterhin gedrückter Maustaste **an eine andere Position** in der Spur gezogen werden. Eine selektierte Nocke kann über die <Entf>-Taste oder den Befehl Entfernen **gelöscht** werden.

Der Nocke kann ein Ereignis zugeordnet werden, welches erfolgen sollen ("switch on", "switch off", "invert", "none") wenn sie in positiver (aufsteigende Master-Werte) bzw. negativer Richtung passiert wird. Sehen Sie unten die Tabelle aller möglichen Kombinationen. Die Attribute der Nocken einer Nockenspur, d.h. mit gleicher "Track ID", beziehen sich auf das selbe Nocken-Bit, eine Variable vom Typ BOOL. Standardmäßig hat eine Nocke die Attribute "switch on" (Einschalten) beim positiven Durchlauf und "switch off" (Ausschalten) beim negativen Durchlauf , so dass die BOOL-Variable beim positiven Durchlauf auf TRUE gesetzt wird und beim negativen auf FALSE. Das Ereignis "invert" bewirkt eine Negation des Werts der BOOL-Variable, die Einstellung "none" hat keine Einfluss.

Die Ein- und Ausschalt-Attribute einer Nocke können geändert werden indem man mit der Maus auf das betreffende Ende der gekreuzten Linien klickt:

Nockensymbol	positiver Durchlauf	negativer Durchlauf
	keine Aktion	keine Aktion
	Umschalten auf ON	keine Aktion
	Umschalten auf OFF	keine Aktion
	keine Aktion	Umschalten auf ON
	keine Aktion	Umschalten auf OFF
	Umschalten auf ON	Umschalten auf OFF
	Umschalten auf ON	Umschalten auf OFF
	Umschalten auf OFF	Umschalten auf ON
	Umschalten auf OFF	Umschalten auf OFF
	Invertieren	keine Aktion
	keine Aktion	Invertieren
	Umschalten auf ON	Invertieren
	Invertieren	Umschalten auf ON
	Invertieren	Umschalten auf OFF
	Umschalten auf OFF	Invertieren
	Invertieren	Invertieren

Registerblatt "Nockentabelle": Tabellarischer Editor für die Definition von Nocken-Tracks. Unterhalb der Zeile mit der Spur-ID des Nockenschalters, der das Symbol vorangestellt ist, folgen soviele Zeilen wie Nocken für diesen Schalter definiert sind. Diesen Zeilen ist jeweils das Symbol vorangestellt und sie enthalten jeweils die Ereignis-Definitionen der Nocke. Es gibt somit die folgenden Tabellenspalten: Track ID (Kennung der Nockenspur); X: Master-Position der Nocke; positive pass / negative pass: Ereignis bei Durchlaufen der Nocke in positiver oder negativer Richtung.

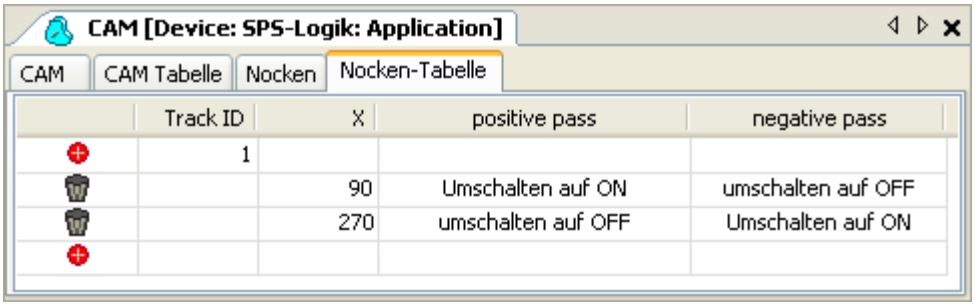


Abb. 302 Haupteditor: Nocken-Tabelle

Eigenschaften von Nocken einer Nockenspur, d. h. derselben Track ID, beziehen sich auf auf dasselbe Nocken-Bit, d.h. auf diesselbe boolsche Variable. Standardmäßig hat eine Nocke die Ereignisdefinition "Umschalten auf ON" bei positivem Durchlauf und "Umschalten auf OFF" bei negativem Durchlauf, so dass diese Variable bei positivem Durchlauf auf TRUE und bei negativem auf FALSE gesetzt wird. Das Ereignis "Invertieren" negiert den Variablenwert, während "keine Aktion" bedeutet, dass sich der Wert nicht verändert.
Um in einer bestehenden Nockenspur eine **Nocke hinzuzufügen** klicken Sie auf das der Track-ID vorangestellte -Symbol. Daraufhin wird unterhalb eine neue Zeile mit Master-Position 0 und den Standard-Ereignis-Attributen eingefügt.

Zum **Ändern eines Tabelleneintrags** müssen Sie diesen mit einem Mausklick selektieren. Dann kann entweder direkt eine neuen Track ID eingegeben werden bzw. für den positiven/negativen Durchlauf von der Auswahlliste das gewünschte Ereignis-Attribut von der Auswahlliste übernommen werden.
Zum Löschen einer Nocke klicken Sie auf das der betreffenden Zeile vorangestellte -Symbol.

Zum **Hinzufügen einer neuen Spur** klicken Sie auf das -Symbol der letzten Tabellenzeile. Daraufhin werden zwei neue Zeilen eingefügt, die erste enthält eine noch nicht verwendete Track ID (Spur-ID), die zweite eine Standard-Nocke mit Master-Position 0 und Standard-Ereignis-Attributen. Eine Spur-Definition wird komplett gelöscht, sobald alle zugehörigen Nocken-Definitionen gelöscht wurden.

Editieren im Elementeigenschaften-Editor

Die **Eigenschaften** eines in einem der graphischen Editoren ausgewählten **Elements** werden im Fenster "Eigenschaften", das standardmäßig neben der Werkzeug-Box erscheint angezeigt und können auch dort bearbeitet werden.

Siehe Elementeigenschaften, um eine detaillierte Beschreibung zu erhalten.

Editieren im Kurvenscheiben-Eigenschaften-Editor

Die Grundeinstellungen eines im Gerätebaum ausgewähltes Kurvenscheiben-Objekts können im **Objekt-Eigenschaften**-Dialog definiert werden.

Siehe Objekt-Eigenschaften, um eine detaillierte Beschreibung dazu zu erhalten.

13.1.5 Kompilieren der Kurvenscheibendefinition

Während des Kompilierens werden für eine Kurvenscheibe Strukturvariablen des Typs **MC_CAM_REF** angelegt. Sie enthalten eine Beschreibung jedes Abbildungssegments der Kurvenscheibe. Datenstrukturen dieser Art werden dem MC_CamIn Modul übergeben. Die Struktur ist Teil der Bibliothek **SM3_ Basic.library**.

13.1.6 Allgemeine Hinweise zu Kurvenscheiben

Die Bibliothek SM3_Basic.library stellt geeignete Module zur Behandlung von Kurvenscheiben zur Verfügung. Sie kann innerhalb des Bibliotheksverwalters mit Hilfe von **Bibliothek hinzufügen...** in das Projekt eingebunden werden. Andererseits wird sie auch automatisch eingebunden, sobald ein Softmotion-Antrieb in den Gerätebaum eingefügt wird.

Die folgenden Abschnitte sollen die Bedeutung bestimmter Parameter (Periodizität, Offset etc.) ausführlich erklären, ebenso wie die Möglichkeit zwischen verschiedenen Kurvenscheiben umzuschalten:

- Effekt der Modulparameter
- Umschalten zwischen Kurvenscheiben

Effekt der Modulparameter

CAM Properties (Properties dialog of the CAM):

- **Periodic transition, Period**

  Eine periodische Kurvenscheibe kann wiederholt abgefahren werden ohne dass beim Übergang von einer zur anderen Periode ein Sprung auftritt. Sofern im Kontrollkästchen vor "Periodic transition" (periodische Transition) ein Häkchen gesetzt ist ☑, ist die der Endposition des Führungsantriebs (Master) zugeordnete Position des Folgeantriebs (Slave) ein Vielfaches der Startposition des Folgeantriebs, wohingegen Geschwindigkeit und Beschleunigung des Slaves an der Endposition exakt mit deren Werte an der Startposition übereinstimmen. Periode und Vorschub werden in der Einheit der Slave Skalierung gemessen.

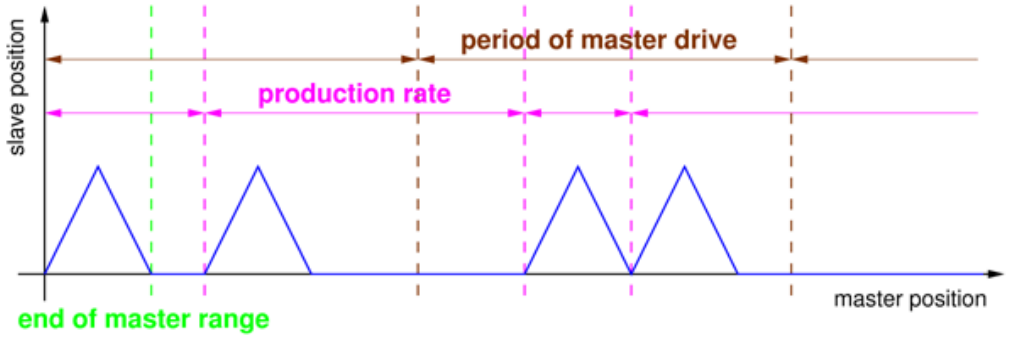
  Selbst wenn "Periodic transition" nicht aktiviert wurde, kann die Kurvenscheibe fortlaufend abgefahren werde. In diesem Fall ist es Aufgabe des Anwenders die Konsistenz der Übergängen in ausreichendem Maß sicher zu stellen.

Modul MC_CAMTableSelect:

- **Periodic**

  Dieser Parameter bestimmt, ob eine Kurvenscheibe nach Erreichen der Endposition des Masters erneut gestartet wird. Das ist der Fall, wenn der Parameter auf TRUE gesetzt wurde.

  Das untenstehende Bild verdeutlicht die Unterschiede der dabei auftretenden Zeitspannen: Als praktischer Anwendungsfall mag ein Fließband gelten, welches identische Objekte befördert. Der Wertebereich des Masters reicht von 0 bis zur Länge dieser Objekte. Innerhalb dieses Bereichs fährt ein Werkzeug an das Objekt heran (bspw. um auf dieses eine Markierung zu stempeln) und verlässt es hierauf wieder. Die Bewegung des Werkzeugs wird vom Folgeantrieb (blauer Graph) gesteuert.



Selbstverständlich ist der Master-Wertebereich der Kurvenscheibe (d. h. der Bereich definierter Positionswerte des Masters) nicht identisch mit der Periode des Führungsantriebs (im Beispiel gerade ein Umlauf des Bands). Daher hat die Zuweisung 'SlavePosition = CAM(MasterPosition)', also die Definition der Slave-Position in Abhängigkeit von der Master-Position mittels der Kurvenscheibe, Gültigkeit nur für das erste Abfahren der Kurvenscheibe.

Bei Eintreffen eines neuen Objekts muss zur Steuerung des Werkzeugs ein neuer Zyklus der Kurvenscheibe gestartet werden. Nachdem die Objekte auf dem Band mit einem gewissen Abstand zueinander platziert sind und dieser Abstand variieren darf, ist die Produktionsrate (d. i. die Zeitspanne zwischen aufeinanderfolgenden Starts der Kurvenscheibe) nicht identisch mit dem Master-Wertebereich der Kurvenscheibe.

Hat der Parameter 'Periodic' den Wert FALSE, so wird bei Erreichen der Endposition des Masters die Ausgangsvariable EndOfProfile des Moduls CamIn auf TRUE gesetzt und der Slave verharrt von da auf seiner aktuellen Position. Beachten Sie, dass durch das Verlassen des Master-Wertebereichs die Aktivität der Kurvenscheibe nicht beendet wird: Bei erneutem Eintritt des Führungsantriebs in den Master-Wertebereich, wird des Folgeantrieb wiederum durch die Kurvenscheibe kontrolliert.

- **MasterAbsolute**

  Ist dieser Eingang TRUE, so wird die Kurvenscheibe an der aktuellen Master-Position gestartet. Dieser Punkt darf auch im Inneren des Master-Wertebereichs der Kurvenscheibe liegen. Ist er jedoch nicht mehr im Wertebereich der Kurvenscheibe enthalten, so wird ein Fehler ausgegeben.

  Ist der Eingang FALSE, so wird die Kurvenscheibe auf die aktuelle Position verlagert, d.h. der Nullpunkt des Masters wird auf die aktuelle Master-Position versetzt. Eine Kurvenscheibe, deren Master-Wertebereich die "0" nicht enthält, darf innerhalb dieses Modus also nicht verwendet werden, da zwangsläufig ein Fehler produziert würde ("...master leaving specified range...").

- **SlaveAbsolute**

  Der Parameter CamTableSelect.SlaveAbsolute beeinflusst den StartMode des Folgeantriebs (der zunächst vom Parameter CamIn.StartMode vorgeschrieben wird). Welcher StartMode aus der Wechselwirkung der beiden Parameter resultiert, ist in nachfolgender Tabelle dokumentiert:

CamIn.StartMode CamTableSelect.SlaveAbsolute	absolute	relative	ramp_in	ramp_in_pos	ramp_in_neg
TRUE	absolute	relative	ramp_in absolute	ramp_in_pos absolute	ramp_in_neg absolute
FALSE	relative	relative	ramp_in relative	ramp_in_pos relative	ramp_in_neg relative

Eine detaillierte Beschreibung der Modi finden Sie unter MC_Camin.StartMode.

Module MC_CamIn

- **MasterOffset, MasterScaling**

  Diese Eingaben transformieren die Master-Position gemäß folgender Formel:

  X = MasterScaling*MasterPosition + MasterOffset

  Die transformierte Position X wird nun zur Auswertung der Kurvenscheibe verwendet. Somit wird die Kurvenscheibe mit einer höheren Geschwindigkeit abgefahren, wenn der Wert von MasterScaling größer ist als 1; die Geschwindigkeit wird hingegen gesenkt für Werte kleiner 1.

- **SlaveOffset, SlaveScaling**

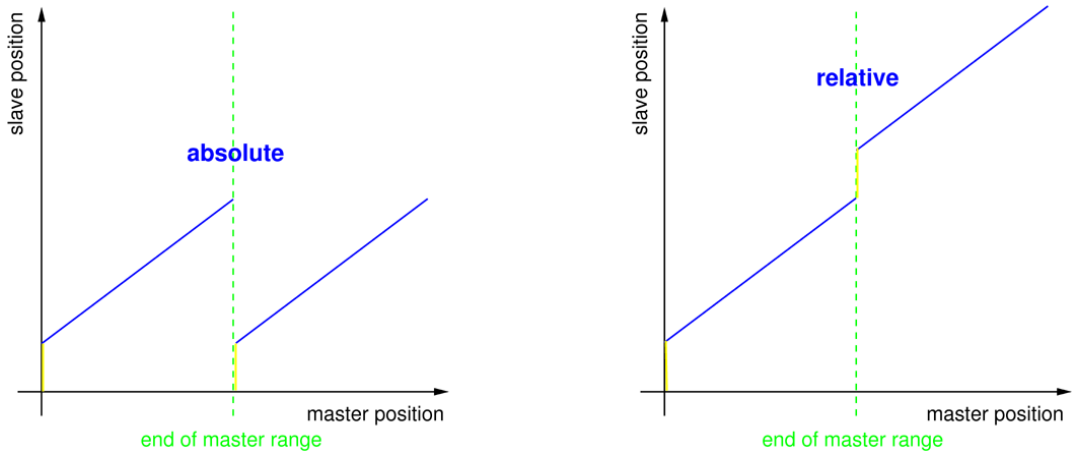
  Diese Eingabe verschieben bzw. strecken den Graph der Kurvenscheibenfunktion in Richtung des Slaves (vertikale Ache). Die Kurvenscheibe wird zunächst gestreckt, anschließend verschoben gemäß folgender Formel:

  Y = SlaveScaling*CAM(X) + SlaveOffset

  Ein SlaveScaling >1 bewirkt ein Strecken der Kurvenscheibe, folglich eine Vergrößerung des Slave-Wertebereichs; entsprechend führt ein SlaveScaling<1 zu einer Stauchung.

StartMode: absolute / relative / ramp_in / ramp_in_pos / ramp_in_neg

Nachstehendes Bild demonstriert den Effekt der Einstellung 'absolute' bzw. 'relative' für eine Beispiel Kurvenscheibe. Der zugehörige Wertebereich des Masters wird links durch die vertikale Achse und rechts durch die grüne gestrichelte Linie beschränkt. Nachdem es sich um eine periodische Kurvenscheibe handelt, wird sie nach Erreichen der Master-Endposition erneut gestartet:



absolute: Beim Start eines neuen Zyklus wir die Kurvenscheibe unabhängig von der aktuellen Position des Slave ausgewertet. Das kann zu einem Sprung führen, falls die zur Startposition des Masters korrelierende Slave-Position abweicht von der zur Endposition des Masters korrelierenden abweicht.

relative: Die neue Kurvenscheibe wird unter Berücksichtigung der aktuellen Slave-Position gestartet, d.h. die Position, die der Slave nach Ende des vorausgehenden Zyklus innehat wird nun als 'slave offset' zu den neuen Auswertungen der Kurvenscheibe addiert. Trotzdem kann noch ein Sprung auftreten sofern die in der Definition der Kurvenscheibe zur Startposition des Masters korrelierende Slave-Position nicht "0" ist

ramp_in: Beim Start der Kurvenscheibe werden eventuell auftretende Sprünge durch ausgleichende Bewegungen (die Beschränkungen für VelocityDiff, Acceleration, Deceleration einhaltend) verhindert. Sofern der Folgeantrieb rotatorisch ist, bewirkt dabei die Option **ramp_in_pos** eine Kompensation allein in positiver Richtung, wohingegen **ramp_in_neg** in negativer Richtung kompensiert. Für lineare Folgeantriebe ergibt sich die Richtung der Kompensation automatisch, **ramp_in_pos** und **ramp_in_neg** werden wie **ramp_in** interpretiert.

Umschalten zwischen Kurvenscheiben

Grundsätzlich ist das Umschalten zwischen verschiedenen Kurvenscheiben jederzeit möglich. Allerdings gibt es einige Punkte, die dabei beachtet werden sollten:
Im Kontext des CAM Editors ist die Position des Slaves eindeutig definiert als Funktionswert der Kurvenscheibenfunktion, welche auf dem Master-Wertebereich definiert ist:

SlavePosition = CAM(MasterPosition)

Da die aktuelle Periode des Masterantriebs i.a. vom Master-Wertebereich abweicht, muss die Master-Position auf den Definitionsbereich der Kurvenscheibenfunktion skaliert werden um ein gültiges Argument darzustellen:

SlavePosition = CAM(MasterScale*MasterPosition + MasterOffset)

In analoger Weise muss der Funktionswert (also die eigentliche Slave-Position) skaliert werden, falls der Start der Kurvenscheibe im Modus 'absolute' zu einem Sprung führen würden:

SlavePosition = SlaveScale*CAM(MasterPosition) + SlaveOffset

Schlimmstenfalls müssen beide Skalierungen angewandt werden, so dass die tatsächliche Slave-Position nach folgender komplexer Formel ermittelt werden muss:

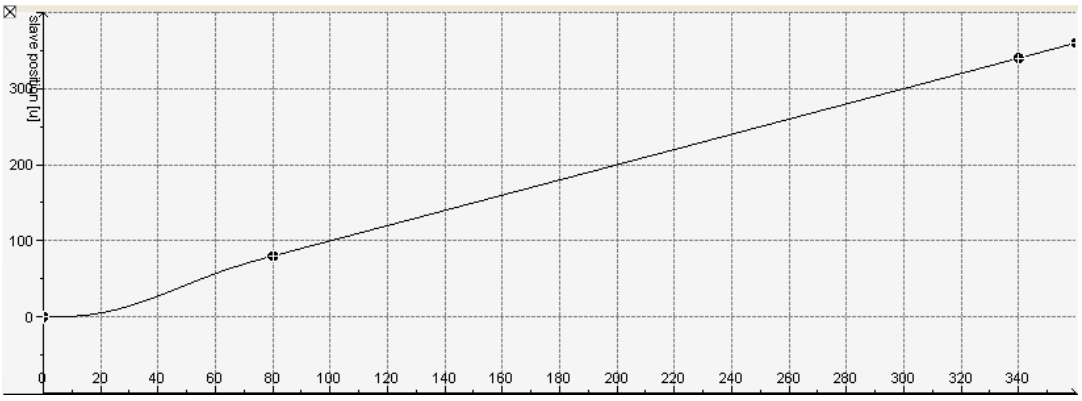
Slaveposition = SlaveScale*CAM(MasterScale*Masterposition + MasterOffset) + SlaveOffset

Die geeigneten Werte für Skalierungs- und Offsetparameter können von Zyklus zu Zyklus variieren.

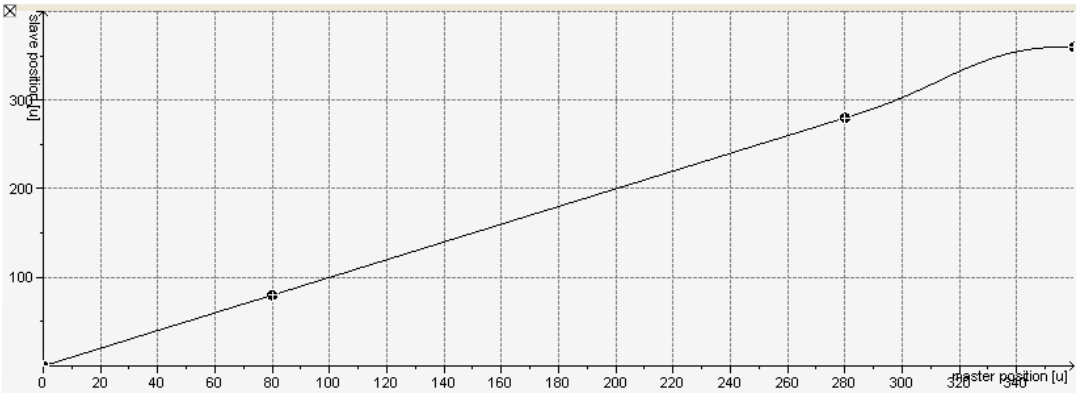
Bedauerlicherweise löscht ein Neustart des Kurvenscheibenmoduls MC_CamIn den zugehörigen Speicherbereich und damit auch die Werte von Skalierung und Offset. Folglich wird die Kurvenscheibenfunktion in ihrer ursprünglichen Definition verwendet, was für gewöhnlich andere Werte für die Slave-Position ergibt. Aus diesem Grund wird empfohlen den Funktionsblock MC_CamIn-FB nur dann neu zu starten, wenn eine andere Kurvenscheibe gestartet werden soll.

Werfen Sie einen Blick auf folgendes Beispiel, in welchem von CAM1 zu CAM2 gewechselt werden soll:

CAM1 besteht aus einem Polynom 5.Grades gefolgt von zwei Geradenstücken.

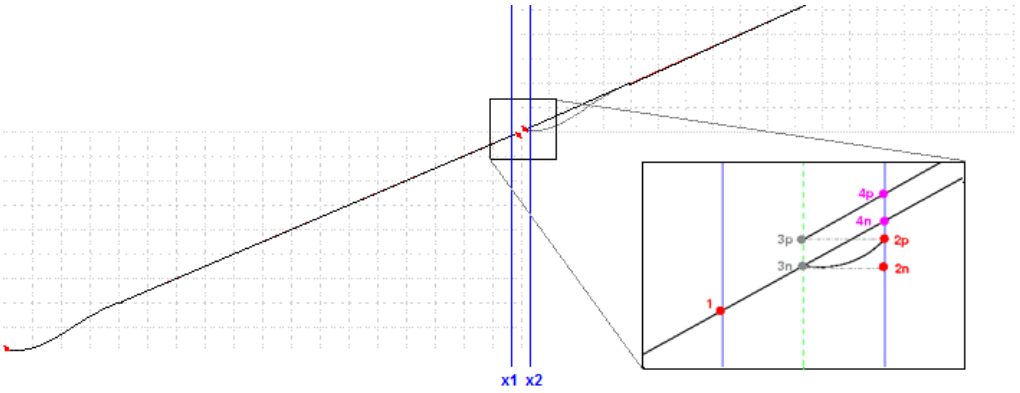


CAM2 besteht aus zwei Geradenstücken gefolgt von einem Polynom 5. Grades.



Beim Umschalten zwischen den beiden Kurvenscheiben sollte man folgendes berücksichtigen:

- Die Werte von Geschwindigkeit und Beschleunigung am Endpunkt der ersten Kurvenscheibe sollten mit den Werten am Anfangspunkt der zweiten Kurvenscheibe übereinstimmen um Sprünge zu vermeiden.
  Im Beispiel ist diese Bedingung erfüllt, da dem Endpunkt von CAM1 wie auch dem Startpunkt von CAM2 dieselbe Geschwindigkeit (=1) und Beschleunigung (=0) zugeordnet ist.
- Die zweite Kurvenscheibe kann im Modus "relative" gestartet werden, falls die Startposition des Slaves mit 0 definiert wurde. Dabei muss die erste Kurvenscheibe jedoch in "non-periodic" Modus laufen. Anderenfalls, wenn CAM1 also periodisch wäre, resultiert die Einstellung "relative" in einem Sprung:



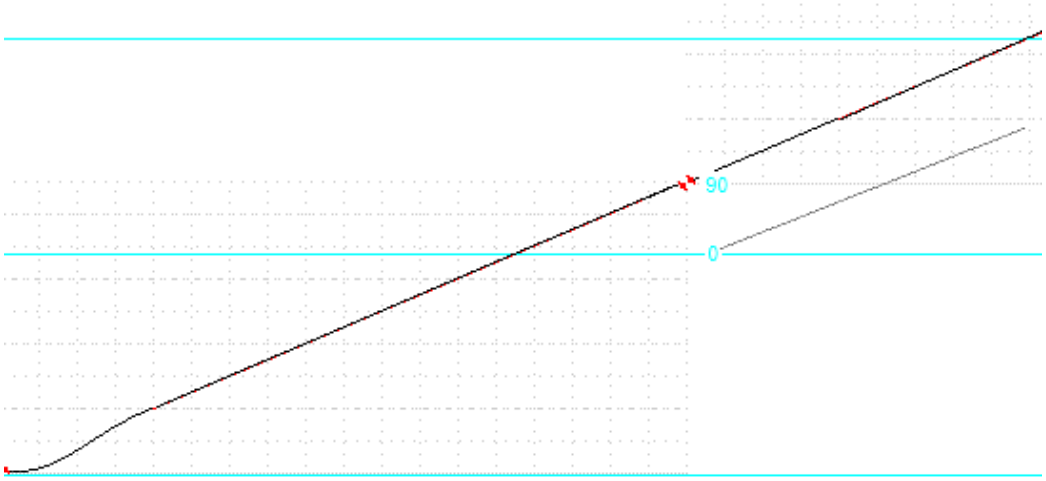
- Der rechte Bildausschnitt ist eine Vergrößerung des Übergangs von CAM1 zu CAM2. Die blauen Linien markieren die Auswertungen der Kurvenscheibenfunktionen an den Master-Positionen x1 und x2. Wir betrachten zunächst den ungünstigen **periodic**Fall:

MasterAbsolute := TRUE; **SlaveAbsolute := FALSE;**	
CAM(x1, CAM1, PERIODIC:=TRUE);	Der Aufruf bedingt eine Auswertung der Kurvenscheibe an der Master-Position x1, die kleiner ist als die Endposition des Masters von CAM1. So erfolgt die Auswertung von CAM1standardmäßig und liefert Punkt 1 als Position für den Slave.
CAM(x2, CAM1, PERIODIC:=TRUE);	Beim folgenden Aufruf des Moduls liegt die Master-Position x2 außerhalb des Master-Wertebereichs von CAM1, dessen Grenze durch die grüne gestrichelte Linie markiert ist und mit der Abszisse des Punktes 3p übereinstimmt. Demzufolge ist das Flag EndOfProfile gesetzt. Da CAM1 im Modus 'periodic' gestartet wurde, erfolgt ihr Neustart am Ende des Wertebereichs, was schließlich den Punkt 2p als Ergebnis des Modulaufrufs liefert.
CAM(EXECUTE:=FALSE);	Wechsel zur neuen Kurvenscheibe.
CAM(x2, CAM2, PERIODIC:=TRUE);	Zweite Auswertung an der Master-Position x2, diesmal wird die neue Kurvenscheibe CAM2 ausgewertet. Nachdem CAM2 im Modus 'relative' gestartet wurde, wird die aktuelle Position (2p) des Slave als Offset zum Bild der Kurvenscheibenfunktion von CAM2 addiert. Dadurch wird der Startpunkt ihres Graphen verschoben auf den Punkt 3p und ihre Auswertung an der Master-Position x2 liefert den Punkt 4p und somit einen unerwünschten Sprung.

- Wählen Sie den Modus **non-periodic** um Sprünge zu vermeiden:

MasterAbsolute := TRUE; **SlaveAbsolute := FALSE;**	
CAM(x1, CAM1, PERIODIC:=FALSE);	Der Ausruf bedingt eine Auswertung der Kurvenscheibe an der Master-Position x1, die kleiner ist als die Endposition des Masters von CAM1. So erfolgt die Auswertung von CAM1standardmäßig und liefert Punkt 1 als Position für den Slave.
CAM(x2, CAM1, PERIODIC:=FALSE);	Beim folgenden Aufruf des Moduls liegt die Master-Position x2 außerhalb des Master-Wertebereichs von CAM1, dessen Grenze durch die grüne gestrichelte Linie markiert ist und mit der Abszisse des Punktes 3n übereinstimmt. Demzufolge ist das Flag EndOfProfile gesetzt. Da CAM1 im Modus 'non-periodic' gestartet wurde, ist die der Master-Position x2 zugeordnete Slave-Position (2n) identisch mit der Position des Slave beim Erreichen des Endes des Wertebereichs von CAM1 (3n).
CAM(EXECUTE:=FALSE);	Wechsel zur neuen Kurvenscheibe.
CAM(x2, CAM2, PERIODIC:=FALSE);	Zweite Auswertung an der Master-Position x2, diesmal wird die neue Kurvenscheibe CAM2 ausgewertet. Nachdem CAM2 im Modus 'relative' gestartet wurde, wird die aktuelle Position (2n) des Slaves als Offset zum Bild der Kurvenscheibenfunktion von CAM2 addiert. Dadurch wird der Startpunkt ihres Graphen verschoben auf den Punkt 3n und ihre Auswertung an der Master-Position x2 liefert den Punkt 4n, welcher auf der durch die Punkte 1 und 3n bestimmten Geraden liegt.

- Sollen die Kurvenscheiben im Modus "absolute" gestartet werden, so muss auf eine passende Startposition des Slaves geachtet werden. Falls der Wertebereich des Masters gerade mit der Periode des Slaves übereinstimmt, wird der Wechsel zwischen den Kurvenscheiben keinerlei Komplikationen mit sich bringen, unabhängig davon, ob die Kurvenscheiben periodisch sind oder nicht.
  In obigem Beispiel darf CAM2 im Modus "absolute" gestartet werden, wenn die Perioden von Master und Slave mit dem Master-Wertebereich von CAM2 übereinstimmen, also jeweils 360° betragen.
  Ansonsten, falls beispielsweise die Periode des Slaves 270° beträgt (angedeutet durch die hellblauen Linien), ist die Option "absolute" nicht zulässig ohne weiter Maßnahmen zu treffen:



  In diesem Fall befindet sich der Slave beim Wechsel von CAM1 zu CAM2 bei 90°. Der Start von CAM2 im Modus 'absolute' bewirkt seinen Sprung auf 0° (angezeigt durch graue Gerade).
  Der Sprung kann allerdings verhindert werden indem man den Offset des Slaves auf den passenden Wert von 90° setzt.

13.1.7 CAM Datenstrukturen

Kurzüberblick Datenstrukturen

Wenn das Projekt kompiliert wird, werden die vorliegenden Kurvenscheiben (CAM) - Daten implizit in eine Liste von **globalen Variablen** konvertiert.

Dabe ist jede Kurvenscheibe durch eine Struktuvariable vom Typ MC_CAM_REF vertreten, die von der IEC-Applikation wie auch von CAM-vorverarbeitenden Funktionen und Funktionsbausteine verwendet werden kann.

Außerdem werden angelegt: die Datenstruktur _SMC_CAM_LIST (ARRAY OF POINTER TO MC_CAM_REF), die sich auf die einzelnen Kurvenscheiben über Pointer bezieht, sowie eine Datenstruktur, die auf alle verfügbaren Kurvenscheiben zeigt.

Die passenden Datentypen werden von der Bibliothek **SM3_Basic.library** bereitgestellt, die automatisch im Bibliotheksverwalter eingefügt wird, sobald ein SoftMotion-Antrieb im Gerätebaum eingehängt wird.

Natürlich kann die Strukturvariable, die eine Kurvenscheibe beschreibt, auch vom IEC-Programm zur Laufzeit erzeugt oder befüllt werden. Deshalb werden im folgenden alle Datenstrukturen, die nicht rein implizit verwendet werden, detaillierter beschrieben:

- MC_CAM_REF
- ARRAY OF SMC_CAMXYVA
- SMC_CAMTable_<type of variables>_<number of elements>_1
- SMC_CAMTable_<variable-type>_<number of elements>_2

Um einen einfachen Zugriff auf die Strukturvariablen zu ermöglichen, werden diese zusammengefasst und in der globalen Variable g_CAMManager mit der Eigenschaft Count und der Methode GetProgram(int n) aufgelistet.

Beispiel: Zugriff auf Datenobjekte der Struktur MC_CAM_REF:

```
PROGRAM CAMManageRef
VAR
  pCAM_Ref: POINTER TO MC_CAM_REF;
  n: INT;
  i: INT;
END_VAR

n := g_CAMManager.Count;
FOR i:=0 TO n-1 DO
pCAM_Ref := g_CAMManager.GetCAM(i);
(* pCAM_Ref verarbeiten *)
END_FOR
```

MC_CAM_REF

Diese Datenstruktur repräsentiert eine generische Kurvenscheibe und enthält die folgenden Komponenten:

wCAMStructID: WORD	hat immer einen festen Wert und wird verwendet um zu prüfen, ob die dem Modul übergebene Datenstruktur eine MC_CAM_REF Struktur ist.
xStart, xEnd : LREAL	Start- und Endposition des Masters. Definition des Bereichs der verfügbaren Master-Werte.
byType : BYTE	Kurvenscheibentyp, d.h. die Art und Weise, wie die Kurvenscheibe repräsentiert wird. Folgende Werte sind möglich: **1**: äquidistanet, 1-dimensionaleTabelle aus Slave-Positionen **2**: nicht-äquidistante, 2-dimensionaleTabelle aus Master/Slave Punktepaaren **3**: polynomiale Beschreibung über einzelne Punkte, bestehend aus Master-Position, Slave-Position, Slave-Geschwindigkeit und Slave-Beschleunigung (XYVA)
byVarType : BYTE (für byType=1 oder byType=2)	Variablentyp, mit dem die Kurventabelle gefüllt wird: **0**: INT **1**: UINT **2**: DINT **3**: UDINT **4**: REAL **5**: LREAL
nElements : INT	Anzahl an Elementen, d.h. je nach Kurvenscheibentyp die Anzahl der Slave-Positionen, Master/Slave-Positionen oder XYVA-Punkte
byInterpolationQuality : BYTE (für byType=1 oder byType=2)	Interpolationsgrad: 1: linear (Standard) oder 3: kubisch
pce: POINTER TO BYTE	Zeiger auf das Datenelement, abhängig vom Typ: **1**: SMC_CAMTable_<VarType>_<nElements>_1 **2**: SMC_CAMTable_<VarType>_<nElements>_2 **3**: ARRAY OF SMC_CAMXYVA
nTappets: INT	Anzahl Nockenschaltaktionen
pt: POINTER TO SMC_CAMTappet	Zeiger auf ARRAY OF SMC_CAMTappet
strCAMName:STRING	Name der Kurvenscheibe

ARRAY OF SMC_CAMXYVA

Eine XYVA-Kurvenscheibe wird durch ein Array von Variablen des Typs SMC_CAMXYVA repräsentiert. Jede Variable beschreibt einen Punkt der Kurvenscheibe über:

dX (Master-Position)

dY (Slave-Position)

dV (erste Ableitung dY/dX; Slave-Geschwindigkeit)

dA (zweite Ableitung d^2Y/dX^2; Slave-Beschleunigung)

Mindestens der Start- und Endpunkt der Kurvenscheibe müssen enthalten sein.

SMC_CAMTable_<variable-type>_<number of elements>_1

Diese Datenstruktur repräsentiert eine äquidistante Kurventabelle. Sie enthält die folgenden Komponenten:

Table: ARRAY [0..<number of elements>-1] OF ARRAY[0..1] OF <type of variables>	Array von Slave-Positionen; mindestens die den Start- und Endpositionen des Masters zugehörigen Slave-Positionen müssen enthalten sein.
fEditorMasterMin: <type of variables>	Das Speichern der Master-Werte in den Tabellen wird skaliert auf den Bereich [fEditorMasterMin, fEditorMasterMax] in SoftMotion Einheiten
fEditorMasterMax: <type of variables>	
fEditoSlaverMin: <type of variables>	Das Speichern der Slave-Werte in den Tabellen wird skaliert auf den Bereich [fEditorSlaveMin, fEditorSLaverMax]] in SoftMotion Einheiten
fEditorSlaveMax: <type of variables>	
fTableMasterMin: <type of variables>	Das Speichern der Master-Werte in den Tabellen wird auf den Bereich [fTableMasterMin, fTableMasterMax] in Tabelleneinheiten skaliert.
fTableMasterMax: <type of variables>	
fTableSlaveMin: <type of variables>	Das Speichern der Slave-Werte in den Tabellen wird auf den Bereich [fTableSlaveMin, fTableSlaveMax] in Tabelleneinheiten skaliert.
fTableSlaveMax: <type of variables>	

SMC_CAMTable_<variable-type>_<number of elements>_2

Diese Datenstruktur repräsentiert eine äquidistante Kurventabelle. Sie enthält die folgenden Komponenten:

Table: ARRAY [0..<number of elements>-1] OF ARRAY[0..1] OF <type of variables>	Array von Master/Slave-Positionspaaren; mindestens die Start- und End-Position müssen enthalten sein Index 0 bezieht sich auf den Master, Index 1 auf den Slave
fEditorMasterMin: <type of variables>	Das Speichern der Master-Werte in den Tabellen wird skaliert auf den Bereich [fEditorMasterMin, fEditorMasterMax] in SoftMotion Einheiten
fEditorMasterMax: <type of variables>	
fEditoSlaverMin: <type of variables>	Das Speichern der Slave-Werte in den Tabellen wird skaliert auf den Bereich [fEditorSlaveMin, fEditorSLaverMax]] in SoftMotion Einheiten
fEditorSlaveMax: <type of variables>	
fTableMasterMin: <type of variables>	Das Speichern der Master-Werte in den Tabellen wird auf den Bereich [fTableMasterMin, fTableMasterMax] in Tabelleneinheiten skaliert.
fTableMasterMax: <type of variables>	
fTableSlaveMin: <type of variables>	Das Speichern der Slave-Werte in den Tabellen wird auf den Bereich [fTableSlaveMin, fTableSlaveMax] in Tabelleneinheiten skaliert.
fTableSlaveMax: <type of variables>	

Beispiel einer manuell erzeugten Kurvenscheibe

Dieses Beispiel zeigt, wie eine Kurvenscheibe im IEC-Programm, d.h. ohne Verwendung des Editors erzeugt werden kann:

Variablen-Deklaration

```
CAM: MC_CAM_REF:=(
byType:=2, (* non-equidistant )
byVarType:=2, (* UINT *)
nElements:=128,
xStart:=0,
xEnd:=360);
Table: SMC_CAMTable_UINT_128_1:=(
fEditorMasterMin := 0, fEditorMasterMax := 360,
fTableMasterMin := 0, fTableMasterMax := 65536,
fEditorSlaveMin := 0, fEditorSlaveMax := 360,
fTableSlaveMin := 0, fTableSlaveMax := 65536);
```

Implementation:

```
(* Kurvenscheibe erzeugen (als Beispiel eine Gerade); unique *)
FOR i:=0 TO 127 DO
Table.Table[i][0]:=Table.Table[i][1]:=REAL_TO_UINT(i / 127.0 * 65536);
END_FOR
(* Zeiger verknüpfen; dies muss in jedem Zyklus erfolgen !!! *)
CAM.pce := ADR(Table);
```

Die so erzeugte Kurvenscheibe kann nun in den Baustein MC_CamTableSelect eingegeben, und dessen Ausgang wieder für MC_CamIn verwendet werden.

13.1.8 CAM: Elementeigenschaften

Die Eigenschaften der Elemente, die im graphischen CAM Editor verwendet werden können im Eigenschaften-Editor angezeigt und bearbeitet werden.

Um das Editorfenster zu öffnen, wählen Sie Befehl **Elementeigenschaften**, standardmäßig im Menü **Ansicht**. Ein Fenster mit dem Titel "Eigenschaften" wird geöffnet. Wenn gerade kein Element (Punkt, Ableitungsabschnitt, Nocke...) im CAM Editor ausgewählt ist, erscheint das Fenster leer und im Feld "Beschreibung" im unteren Teil wird nichts dargestellt.

Wenn jedoch in einem der graphischen Registerblätter des CAM Editors ("CAM" oder "Tappets") ein Element mit einem Mausklick ausgewählt ist, werden die entsprechenden Eigenschaften in Tabellenform dargestellt.

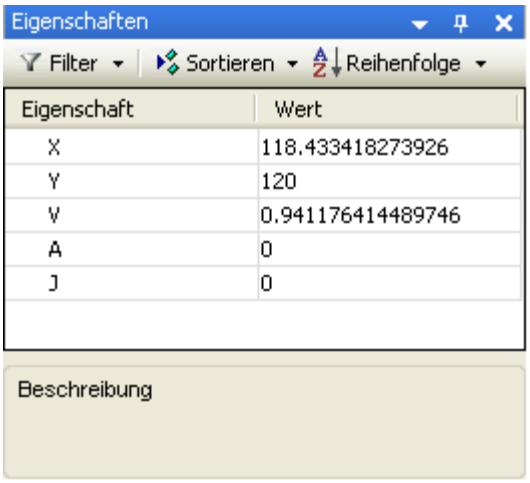


Abb. 303 Beispiel des Eigenschaften-Editors, Anzeige der Eigenschaften eines Punktes

Der Editor bietet eine kleine Werkzeugleiste am oberen Rand. Das erste davon kann leider noch nicht genützt werden: Es gibt noch keine Filter-Funktionen, es werden immer alle Eigenschaften eines Elements angezeigt. Mit dem zweiten Werkzeug **Sortieren** kann die Anzeige der Eigenschaften wahlweise nach Typ oder alphabetisch sortiert werden. Wählen Sie die gewünschte Einstellung mit einem Mausklick.

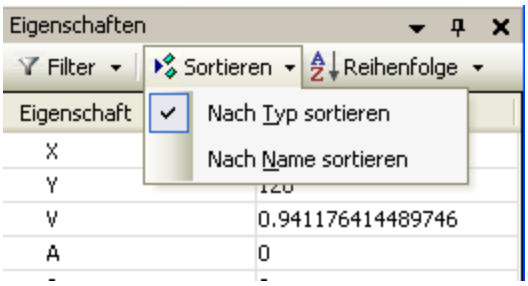


Abb. 304 Werkzeugleiste: Sortieren...

Das dritte Werkzeug dient dazu, die Anzeige der Eigenschaften in der Reihenfolge umzukehren, d.h. in aufsteigender oder absteigender Reihenfolge darzustellen. Wählen Sie das Gewünschte durch Mausklick.

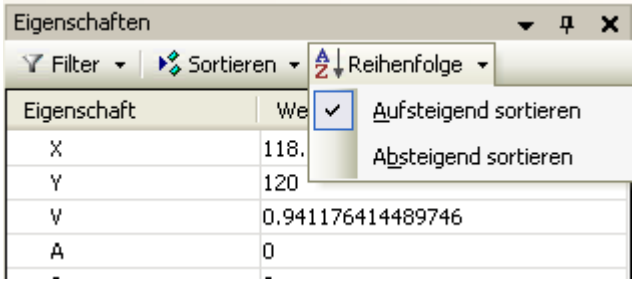


Abb. 305 Werkzeugleiste: Reihenfolge

Ein Tabelleneintrag wird durch einen Mausklick selektiert. Die Selektion wird durch einen dünnen blauen Rahmen und eine hellblaue Schattierung der Tabellenzeile angezeigt. Gleichzeitig wird die entsprechende Beschreibung der Eigenschaft im Feld unterhalb der Tabelle angezeigt.

Abhängig von der Eigenschaft kann der **Wert** des selektierten Eintrags durch Tippen oder aus einer Auswahlliste **modifiziert** werden, nachdem ein weiterer Mausklick auf die Tabellenzelle ausgeführt wurde. Der geänderte Wert wird übernommen, sobald die Tabellenzelle wieder de-selektiert ist, d.h. nach einem Mausklick auf eine andere Tabellenzelle oder auf leeren Raum im Editor.

13.2 CFC-Editor

Der CFC Editor, der mit dem Plug-In 'CFC Editor' installiert wird, dient dem Programmieren von Objekten in der Sprache **Continuous Function Chart** (CFC), die in Erweiterung der Standard-Sprachen der IEC 61131-3 zur Verfügung steht. Die Programmiersprache wählen Sie beim Anlegen eines neuen Objekts 'Objekt hinzufügen'-Dialog aus.

Der CFC Editor ist ein grafischer Editor.

Der Editor befindet sich im unteren Teil des Fensters, das sich öffnet, wenn ein CFC Objekt bearbeitet wird, und das im oberen Teil auch den Deklarationseditor enthält.

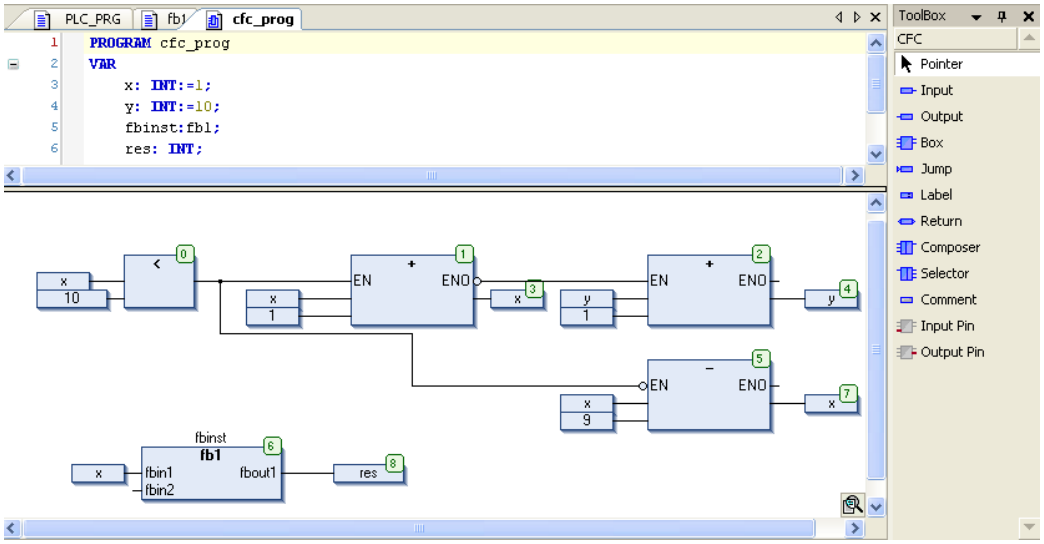


Abb. 306 CFC Editor

Der CFC Editor erlaubt im Gegensatz zu den netzwerk-basierenden Editoren ein freies Positionieren der Elemente, was beispielsweise das direkte Einfügen von Rückkopplungen ermöglicht. Die Abarbeitungsreihenfolge wird durch eine Liste der aktuell eingefügten Elemente bestimmt, die auch verändert werden kann.

Die folgenden Elemente stehen in einer Werkzeugbox zum Einfügen bereit: Baustein (Operatoren, Funktionen, Funktionsblöcke, Programme), Eingang, Ausgang, Kommentar, Marke, Sprung, Komposition, Selektor.

Die Ein- und Ausgangs-Pins der Elemente können verknüpft werden indem mit der Maus eine Verbindungslinie gezogen wird. Die Verbindungslinie wird automatisch so angelegt, dass sie den kürzesten möglichen Weg zwischen den Elementen zurücklegt. Die Verbindunglinien werden außerdem automatisch angepaßt, sobald ein Element verschoben wird. Sehen Sie hierzu auch: Elemente einfügen und anordnen.

Die Größe des Editor ist veränderbar: Benützen Sie hierzu die Taste in der unteren rechten Ecke des Fenster und wählen Sie einen der in der Liste aufgeführten Vergrößerungsfaktoren. Alternativ wählen Sie den Listeneintrag **...** um einen beliebigen Faktor einzugeben.

Die Befehle für das Arbeiten im CFC Editor können aus dem Kontextmenü oder aus dem "CFC" Menü aufgerufen werden, das standardmäßig verfügbar ist, wenn der CFC Editor aktiv ist.

13.2.1 CFC - Continuous Function Chart - Sprache

Das "Continuous Function Chart" ist eine grafische Programmiersprache, die in Erweiterung der IEC61131-3 Standardsprachen auf der Funktionsplan-Sprache (FUP) basiert. Im Gegensatz zu dieser jedoch erlaubt sie das freie Positionieren von Elementen, was beispielsweise ermöglicht, direkt Rückkopplungen einzufügen.

Zum Erzeugen von CFC-Programmbausteinen in »PLC Designer« siehe: CFC Editor.

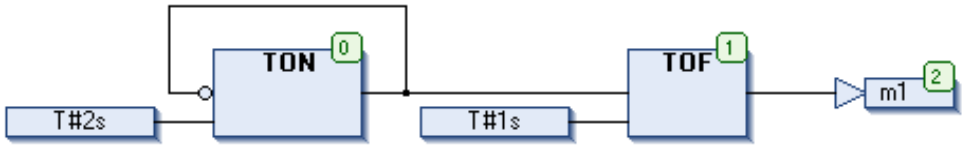


Abb. 307 Beispiel eines CFC Netzwerks

13.2.2 Cursor-Positionen in CFC

Eine mögliche Cursor-Position im CFC Editor wird standardmäßig durch eine graue Schattierung angezeigt, wenn man den Cursor über die eingefügten Elemente bewegt.

Wenn man auf eine dieser schattierten Bereiche klickt und die Maustaste noch gedrückt hält, verändert der Bereich seine Farbe nach Rot. Sobald die Maustaste losgelassen wird, wird die Stelle zur aktuellen Cursor-Position, d.h. das jeweilige Element wird ausgewählt und in roter Farbe dargestellt.

Es gibt drei Kategorien von Cursor-Positionen. Sehen Sie im folgenden die möglichen, aber noch nicht eingenommenen Positionen jeweils durch die graue Schattierung angezeigt:

1. Wenn der Cursor auf einem **Text** platziert wird, wird dieser blau hinterlegt dargestellt und kann editiert werden. Die [...]-Schaltfläche steht zum Öffnen der Eingabehilfe bereit. Nach dem Einfügen des Elements repräsentieren zunächst drei Fragezeichen "???" den Text und müssen durch einen gültigen Bezeichner ersetzt werden. Hinterher wird bei Positionierung des Cursors auf den Namen einer Variablen oder eines Box-Parameters ein Tooltip angezeigt, der den Typ der Variablen/des Parameters enthält und, sofern vorhanden, den zugehörige Kommentar in einer zweiten Zeile.

Abb. 308 Mögliche Cursor-Positionen und zwei Beispiele von ausgewähltem Text im CFC

2. Wenn der Cursor auf dem **Rumpf eines Elements** (Baustein, Eingang, Ausgang, Kommentar, Marke, Sprung, Komposition, Selektor) platziert wird, wird dieser rot dargestellt und kann mit Hilfe der Maus verschoben werden.

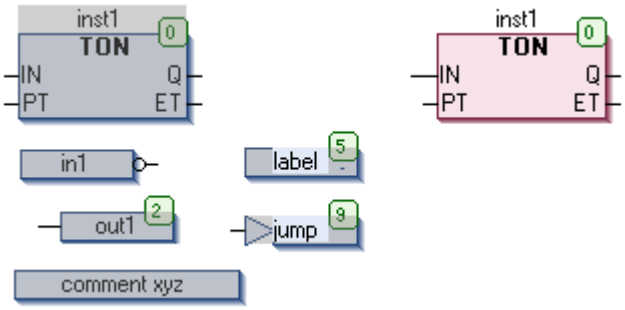


Abb. 309 Mögliche Cursor-Positionen und Beispiel eines ausgewählten Bausteins

3. Wenn der Cursor auf einem **Eingangs**- oder **Ausgangs**pin eines Elements platziert wird, wird dieser rot dargestellt und kann negiert oder gesetzt bzw. zurückgesetzt werden.

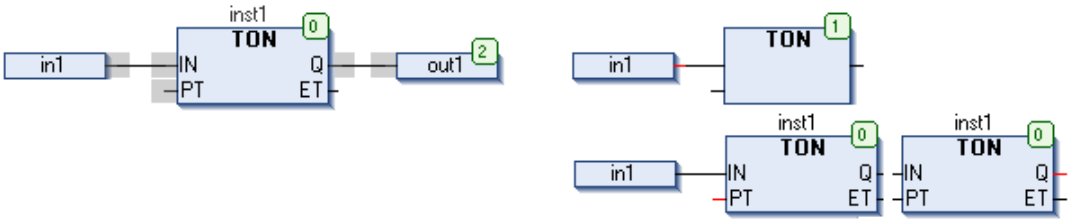


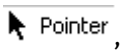
Abb. 310 Mögliche Cursor-Positionen und Beispiele von ausgewählten Ausgangs- und Eingangs-Pins

13.2.3 CFC Elemente / Werkzeuge

Die grafischen Elemente zur Programmierung im CFC Editor werden in einer "Werkzeug-Box" bereitgestellt, die mit dem 'CFC Editor' Plug-In geliefert wird. Diese wird normalerweise automatisch in einem Fenster geöffnet, wenn der CFC-Editor aktiv ist, kann aber standardmäßig auch über den Befehl 'Werkzeuge' (Menü 'Ansicht') explizit geöffnet werden.

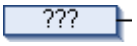
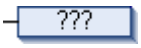
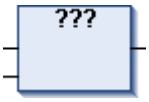
Abb. 311 CFC Werkzeuge, Standardausstattung

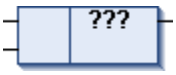
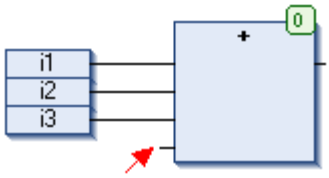
Das gewünschte Element wird in der Werkzeug-Box ausgewählt und über Drag&Drop im Editorfenster eingefügt.

Neben den Programmier-Elementen gibt es auch einen Eintrag 

, normalerweise an oberster Stelle der Werkzeugliste. Solange dieser Eintrag angewählt ist, hat der Cursor die Form eines Pfeils und im Editorfenster können bereits eingefügte Elemente ausgewählt werden um sie zu positionieren und zu bearbeiten.

Die CFC-Elemente:

	Eingang		Der Text, der durch "???" repräsentiert wird, kann ausgewählt und durch einen Variablennamen oder eine Konstante ersetzt werden. Die Eingabehilfe kann verwendet werden, um einen gültigen Bezeichner auszuwählen.
	Ausgang		
	Baustein		Ein Baustein-Element wird verwendet, um einen Operator, eine Funktion, einen Funktionsblock oder ein Programm einzufügen. Die "???" müssen durch den Namen des gewünschten Operators etc. ersetzt werden. Die Eingabehilfe unterstützt dabei das Auswählen eines verfügbaren Objekts. Im Fall eines Funktionsblocks werden zusätzlich auch oberhalb des Baustein-Symbols "???" angezeigt, die durch den Namen der Funktionsblock-Instanz ersetzt werden müssen. Wenn der Baustein anstelle eines bereits vorhandenen eingefügt wird (der vorhandene wird ausgewählt und dann der neue eingefügt) und eine andere Mindestanzahl an Eingängen hat, werden diese automatisch hinzugefügt. Wenn der neue Baustein eine kleinere Höchstzahl an Eingängen hat, werden die letzten Eingänge des vorher vorhandenen Elements gelöscht.
	Sprung		Das Sprung-Element wird verwendet um eine Position festzulegen, an der die Programmabarbeitung fortsetzen soll. Diese Position muss durch eine "Marke" definiert sein (s.u.). Der Text "???" muss also mit dem Namen der Marke ersetzt werden.
	Marke		Eine Marke zeigt eine Position an, zu der die Programmabarbeitung mit Hilfe eines Sprung-Elements (s.o.) springen kann. Im Online-Modus wird automatisch eine RETURN-Marke am Ende eines CFC-Bausteins.
	Return		Beachten Sie, dass im Online-Modus im CFC-Editor ein RETURN- Element automatisch vor der ersten Zeile und nach dem letzten Element Bausteins eingefügt wird. Bei der schrittweisen Abarbeitung wird das RETURN-Element am Ende automatisch angesprungen bevor der Baustein verlassen wird.

	Komposition		Ein Kompositions-Element dient dazu, Eingänge, die von einem Baustein vom Typ einer Struktur kommen, zu handhaben. Der Kompositor stellt die Strukturkomponenten dar und macht sie dem Programmierer im CFC zugänglich. Zu diesem Zweck muss das Kompositor-Element wie die betreffende Struktur genannt werden (Ersetzen der "???") und anstelle eines Eingang-Elements mit dem Baustein verbunden werden.
	Selektor		Ein Selektor-Element wird im Gegensatz zum Kompositions-Element verwendet, um den Ausgang eines Bausteins zu handhaben, der vom Typ einer Struktur ist. Der Selektor stellt die Strukturkomponenten dar und macht sie dem Programmierer im CFC Editor zugänglich. Zu diesem Zweck muss der Selektor wie die betreffende Struktur genannt werden (Ersetzen der "???") und anstelle eines Ausgangs-Elements mit dem Bausteins verbunden werden.
	Kommentar		Mit diesem Element kann ein Kommentar in das CFC eingefügt werden. Ersetzen Sie den Platzhaltertext im Element durch den Kommentartext. Ein Zeilenumbruch kann mit Hilfe der Tastenkombination <Strg> + <Eingabe> eingefügt werden.
	Bausteineingang		Abhängig vom Bausteintyp können einem eingefügten Baustein-Element weitere Eingänge hinzugefügt werden. Dazu muss das Baustein-Element selektiert werden und das Bausteineingang-Element auf den Baustein-Rumpf gezogen werden.
	Bausteinausgang		Abhängig vom Bausteintyp können einem eingefügten Baustein-Element weitere Ausgänge hinzugefügt werden. Dazu muss das Baustein-Element selektiert werden und das Bausteinausgang-Element auf den Baustein-Rumpf gezogen werden.

Beispiel für Komposition

Ein CFC Programm *cfc_prog* verwendet eine Instanz von Funktionsblock *fubblo1*, die eine Eingangsvariable *struvar* vom Typ einer Struktur hat. Durch Verwendung des Kompositions-elements können die Strukturelemente angesprochen werden:

Definition von Struktur *stru1*:

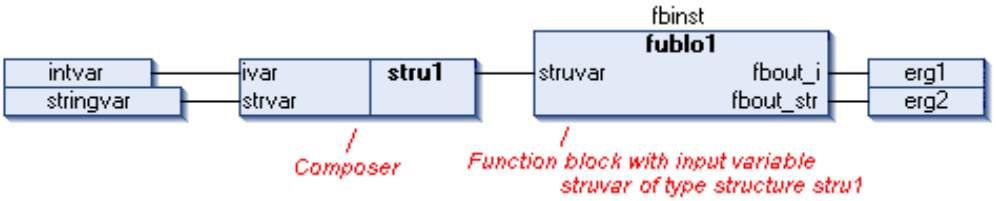
```
TYPE stru1 :
STRUCT
 ivar:INT;
 strvar:STRING:='hallo';
END_STRUCT
END_TYPE
```

Funktionsblock *fublo1*, Deklaration und Implementation:

```
FUNCTION_BLOCK fublo1
VAR_INPUT
 struvar:STRU1;
END_VAR
VAR_OUTPUT
 fbout_i:INT;
 fbout_str:STRING;
END_VAR
VAR
 fbvar:STRING:='world';
END_VAR

fbout_i:=struvar.ivar+2;
fbout_str:=CONCAT (struvar.strvar,fbvar);
```

CFC-Programm *cfc_prog*, Deklaration und Implementation:

```
PROGRAM cfc_prog
VAR
 intvar: INT;
 stringvar: STRING;
 fbinst: fublo1;
 erg1: INT;
 erg2: STRING;
END_VAR
```



Beispiel für Selektor

Ein CFC-Programm *cfc_prog* verwendet eine Instanz von Funktionsblock *fubblo2*, der eine Ausgangsvariable *fbout* vom Typ der Struktur *stru1* hat. Durch Verwenden des Selektor-Elements kann auf die Strukturelemente zugegriffen werden:

Definition der Struktur *stru1*:

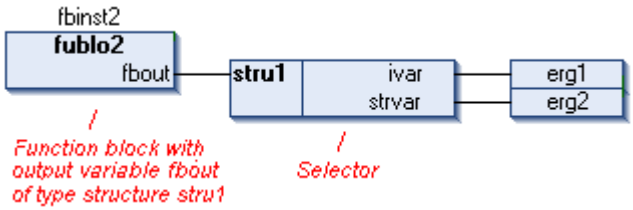
```
TYPE stru1 :
STRUCT
 ivar:INT;
 strvar:STRING:='hallo';
END_STRUCT
END_TYPE
```

Funktionsblock *fublo1*, Deklaration und Implementation:

```
FUNCTION_BLOCK fublo2
VAR_INPUT CONSTANT
 fbin1:INT;
 fbin2:DWORD:=24354333;
 fbin3:STRING:='hallo';
END_VAR
VAR_INPUT
 fbin : INT;
END_VAR
VAR_OUTPUT
 fbout : stru1;
 fbout2:DWORD;
END_VAR
VAR
 fbvar:INT;
 fbvar2:STRING;
END_VAR
```

Programm *cfc_prog*, Deklaration und Implementation:

```
VAR
 intvar: INT;
 stringvar: STRING;
 fbinst: fublo1;
 erg1: INT;
 erg2: STRING;
 fbinst2: fublo2;
END_VAR
```



13.2.4 Elemente einfügen und anordnen

Die Elemente zum Programmieren im CFC Editor werden in der Werkzeugbox bereitgestellt, die standardmäßig in einem Fenster geöffnet ist, wenn der CFC Editor aktiv ist.

Die CFC Editor Optionen definieren generelle Einstellungen für das Arbeiten im Editor.

Einfügen

Um ein Element einzufügen, wählen Sie es mit einem Mausklick im Werkzeuge-Fenster aus, halten die Maustaste gedrückt und ziehen es in das Editor-Fenster. Während des Ziehens wird der Cursor als Pfeil mit angehängtem Rechteck und einem Pluszeichen dargestellt. Nach Loslassen der Maustaste wird das Element eingefügt.

Selektieren

Um ein Element auszuwählen (selektieren), um es bearbeiten oder verschieben etc. zu können, beachten Sie die möglichen Cursor-Positionen im Element-Rumpf, auf den Ein- oder Ausgängen oder im Text. Durch einen Mausklick auf den Rumpf wird das Element selektiert und per Default rot-schattiert dargestellt. Wenn zusätzlich die <Umsch>-Taste gedrückt wird, können weitere Elemente angeklickt und zusätzlich selektiert werden. Alternativ kann die linke Maustaste gedrückt werden und ein gepunkteter Rahmen und diejenigen Elemente gezogen werden, die ausgewählt werden sollen. Die Elemente werden als "ausgewählt" dargestellt, sobald die Maustaste losgelassen wird. Mit dem Befehl Alle selektieren, der standardmäßig im Kontextmenü liegt, können alle Elemente gleichzeitig ausgewählt werden.
Mit den Pfeiltasten kann die Selektionsmarkierung zur nächsten möglichen Cursor-Position verschoben werden. Die Reihenfolge hängt von der Abarbeitungsreihenfolge der Elemente ab, die durch die Elementnummern angezeigt wird, siehe unten.
Wenn ein Eingangs-Pin selektiert ist und <Strg>+<Linkspfeil> gedrückt wird, wird der zugehörige Ausgangs-Pin selektiert. Wenn ein Ausgangs-Pin selektiert ist und <Strg>+<Linkspfeil> gedrückt wird, wird der zugehörige Ausgang (können auch mehrere sein) selektiert.

Verschieben

Um ein Element zu verschieben, wählen Sie es mit einem Mausklick auf den Rumpf aus (siehe mögliche Cursor-Positionen) und ziehen Sie es bei weiterhin gedrückter Maustaste zur gewünschten Position. Nach Loslassen der Maustaste wird das Element dort platziert. Alternativ können die Befehle Ausschneiden und Einfügen verwendet werden.

Verknüpfen

Verbindungslinien zwischen Aus- und Eingängen von Elementen können mit der Maus gezogen werden. Die kürzeste mögliche Verbindung wird angelegt, unter Berücksichtigung der bereits vorhandenen Elemente und Verbindungslinien. Wenn eine Verbindungslinie hellgrau angezeigt wird, überdeckt sie sich mit einer anderen.

Kopieren

Um ein Element zu kopieren, wählen Sie es aus und verwenden die Befehle Kopieren und Einfügen.

Text editieren

Nach dem Einfügen eines Elements wird der Textteil durch "???" repräsentiert. Um diesen Platzhalter durch einen gültigen Bezeichner (Name einer POU, einer Marke, Instanz etc.) wählen Sie den Text mit einem Mausklick an, um ein Eingabefeld zu öffnen. Auch die Schaltfläche … steht dann bereit um die Eingabehilfe zu öffnen.

Löschen

Ein ausgewähltes Element kann mit dem Befehl Löschen, der standardmäßig im Kontextmenü verfügbar ist, oder mit Hilfe der <Enf>-Taste entfernt werden,

Ausführungsreihenfolge, Elementnummern

Die Reihenfolge, in welcher die Elemente eines CFC Netzwerks im Online-Modus nacheinander ausgeführt werden, wird durch die Elementnummern in der oberen rechten Ecke eines jeden Baustein-, Ausgang-, Sprung-, Return- und Marke-Elements angezeigt. Die Abarbeitung startet beim Element mit der niedrigsten Nummer, "0".

Die Ausführungsreihenfolge kann über Befehle verändert werden, die standardmäßig im Untermenü "Ausführungsreihenfolge" des CFC Menüs enthalten sind. Grundsätzlich ist eine Ausführungsreihenfolge nach topologischer Anordnung oder nach Datenfluss eingestellt werden.

Beim Einfügen eines Elements wird die Nummer automatisch in topologischer Reihenfolge vergeben (von links nach rechts und von oben nach unten). Wurde die Reihenfolge bereits verändert, erhält das neue Element die Nummer seines topologischen Nachfolgers und alle höheren Nummern werden um eins erhöht.

Beim Verschieben eines Elementes bleibt die Nummer erhalten.

Bedenken Sie, dass die Reihenfolge Einfluß auf das Ergebnis hat und in bestimmten Fällen geändert werden muss.

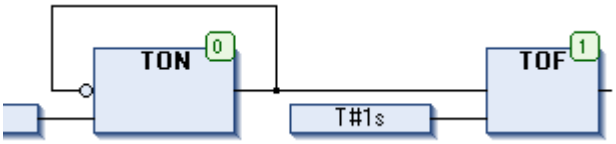


Abb. 312 Elementnummerierung, Beispiel

Größe des Arbeitsblatts im Editorfenster ändern

Um mehr Fläche um ein bestehendes CFC Diagramm im Editorfenster zu erhalten, kann der Arbeitsbereich ("Arbeitsblatt") in der Größe verändert werden. Dies geschieht entweder durch Auswählen und Verschieben aller Elemente mit der Maus, oder mit Hilfe der Befehle Ausschneiden und Einfügen (siehe oben, Verschieben).

Alternativ kann ein spezieller Dialog verwendet werden, um die Ausmaße des Arbeitsblatts festzulegen. Dies kann im Falle von sehr großen Diagrammen sinnvoll sein. Sehen Sie dazu 'Arbeitsblatt bearbeiten'.

13.2.5 CFC Editor im Online-Modus

Im Online-Modus bietet der CFC Editor Ansichten zum Anzeigen (Monitoring) und zum Schreiben und Forcen von Variablenwerten aus der Steuerung. Debugging-Funktionalität (Haltepunkte, Schrittweises Ausführen etc.) ist verfügbar. Siehe unten.

Beachten Sie, dass das Editorfenster eines ST-Objekts auch den Deklarationseditor im oberen Teil enthält.

Sehen Sie Benutzerobefläche im Online-Modus für eine Beschreibung, wie Objekte im Online Modus geöffnet werden.

- Beachten Sie, dass das Editorfenster eines ST-Objekts auch den Deklarationseditor im oberen Teil enthält. Sehen Sie hierzu auch Deklarationseditor im Online-Modus.

Monitoring

Der aktuelle Variablenwert wird in einem kleinen Fenster hinter der Variable angezeigt (Inline-Monitoring).

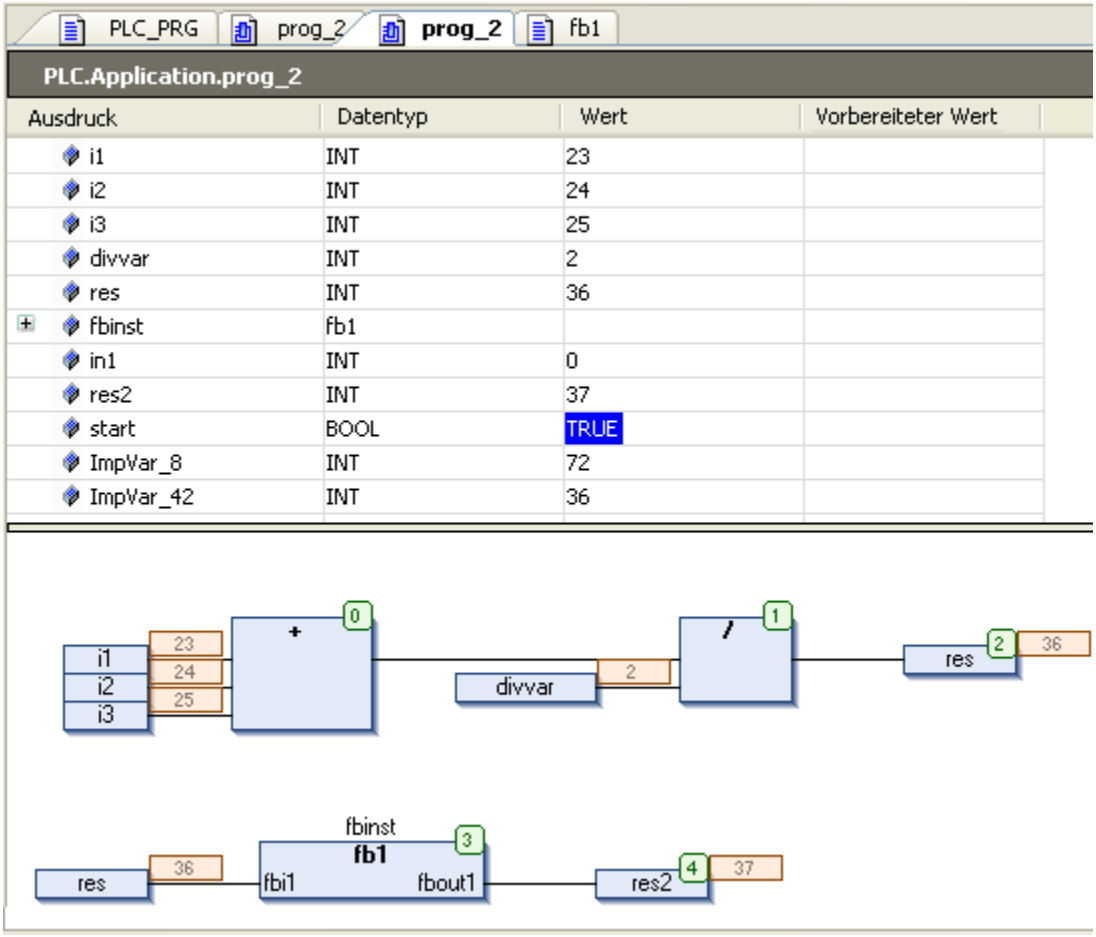


Abb. 313 Beispiel: Online-Ansicht eines Programms PLC_PRG

Beachten Sie für die Online-Ansicht eines Funktionsblocks, dass nur in der Ansicht einer Instanz Monitoring möglich ist, in der Ansicht der Basis-Implementation werden keine Werte angezeigt. Anstatt enthält die Spalte 'Wert' den Text "<Der Wert des Ausdrucks kann nicht gelesen werden>" und in den Inline-Monitoring-Feldern im Implementationsteil stehen jeweils drei Fragezeichen.

Haltepunkt-Positonen im CFC Editor:

Der Anwender kann einen Haltepunkt (Breakpoint) grundsätzlich an den Positionen eines Bausteins setzen, an denen sich ein Variablenwert ändern kann oder wo der Programmverlauf sich verzweigt oder ein anderer Baustein aufgerufen wird. In der folgenden Abbildung zeigen die roten Kreise mögliche Haltepunkt-Positionen an:

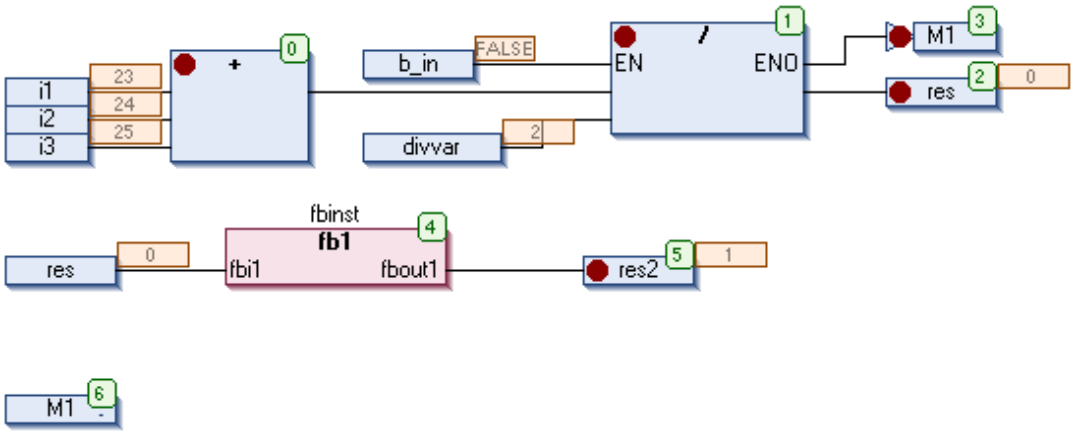


Abb. 314 Haltepunkt-Positionen im CFC Editor

Hinweis für Haltepunkte in Methoden: Ein Haltepunkt wird automatisch in allen Methoden gesetzt, die aufgerufen werden können. Somit gilt: Wenn eine Methode aufgerufen wird, die über eine Schnittstelle definiert ist, werden in allen Methoden von Funktionsblöcken, die diese Schnittstelle implementieren, Haltepunkte gesetzt und ebenfalls in allen abgeleiteten Funktionsblöcken, die die Methode definieren.

13.3 CNC-Editor

Der CNC-Editor ist integriert in »PLC Designer«. Innerhalb des CNC-Editors können komplexe mehrdimensionale Bewegungen im grafischen und Text- bzw. tabellarischen Editor implementiert werden. Die Eigenschaften von Bahnelementen können in einem Fenster betrachtet werden und die Werkzeuge ermöglichen eine einfache grafische Programmierung von CNC-Bahnen.

Auf dieser Seite ist folgendes zu finden:

- Überblick
- Erzeugen eines CNC Programms
- Editieren eines CNC Programms
- Kompilieren des CNC Programms
- Öffnen eines existierenden Programms

Hinweis: Beim Verwenden der F1 Taste öffnet direkt die Online Hilfe mit der Hilfeseite des aktiven Dialogs oder Editors.

Überblick

Der CNC-Editor in »PLC Designer« ermöglicht es, mehrdimensionale Bahnen simultan grafisch und textlich nach der CNC-Sprache DIN66025 zu programmieren. Statt dem Texteditor kann ein Tabelleneditor benutzt werden.

Beachten Sie dazu die erläuterten Programmierbeispiele.

In DIN66025 können grundsätzlich bis zu 9-dimensionale Bahnen realisiert werden, wobei drei Dimensionen nicht linear interpoliert werden. Auf diese Weise können in drei Dimensionen Linien, Kreise, Kreisbögen, Parabeln und Splines programmiert werden. Die weiteren Richtungen werden lediglich linear interpoliert. Für jede programmierte Bahn erzeugt »PLC Designer« automatisch eine globale Datenstruktur (CNC Daten), die in einem IEC-Programm verwendet werden kann.

Dies kann auf verschiedene Weise geschehen:

1. Das CNC-Programm wird als Array aus G-Code-Worten abgelegt, die zur Laufzeit des SPS-Programms mit Hilfe eines Decoder-Bausteins entschlüsselt werden, so dass für die einzelnen Bahnobjekte GEOINFO-Strukturobjekte zur Verfügung stehen. Diese können dann mit Hilfe von Bahnvorverarbeitungsmodulen (siehe Bibliothek SM3_CNC), wie z.B. Werkzeugradiuskorrektur, bearbeitet und anschließend interpoliert, transformiert und dem Drive-Interface wieder zur Kommunikation an die Hardware übergeben werden.
2. Das CNC-Programm wird als Liste von GEOINFO-Strukturobjekten mit Namen SMC_OUTQUEUE in eine Datenstruktur geschrieben und kann so direkt in den Interpolator eingegeben werden. Im Vergleich zu 1. spart man da durch den Aufruf des Decoder-Funktionsbausteins und der Bahnvorverarbeitungs-Funktionsbausteine. Aber man hat dagegen nicht die Möglichkeit, das Programm zur Laufzeit zu verändern.

3. Das CNC-Programm wird als ASCII-Datei in das Dateisystem der Steuerung geschrieben und zur Laufzeit Schritt für Schritt ausgelesen und umgesetzt. Diese Methode eignet sich vor allem für große Programme, die nicht komplett im Speicher gehalten werden können.

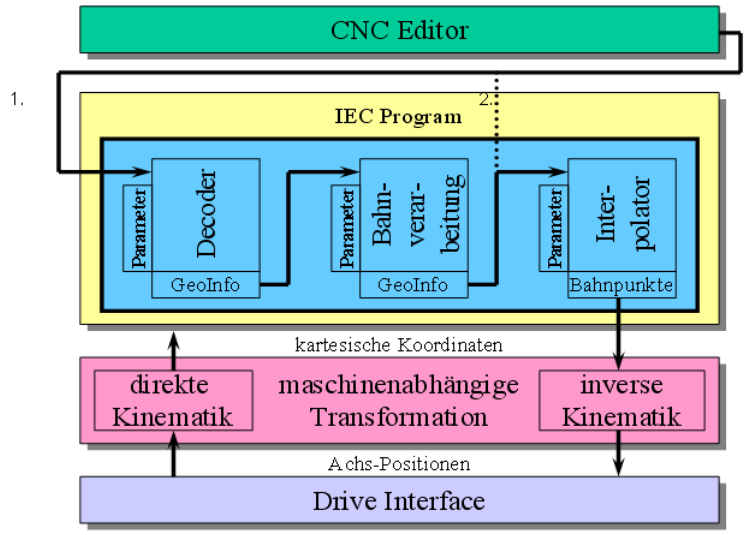


Abb. 315 Die SoftMotion-Software-Bestandteile des CNC Editors

Erzeugen eines CNC-Programms

Ein CNC-Progamm ist ein Objekt, das einer Applikation hinzugefügt wird. Dazu wählt man eine Applikation und aktiviert den Eintrag **Objekt hinzufügen** des Kontextmenüs. Aus der Liste von Objekten wird **CNC-Programm ...** gewählt. Der Dialog **CNC-Programm hinzufügen** öffnet.

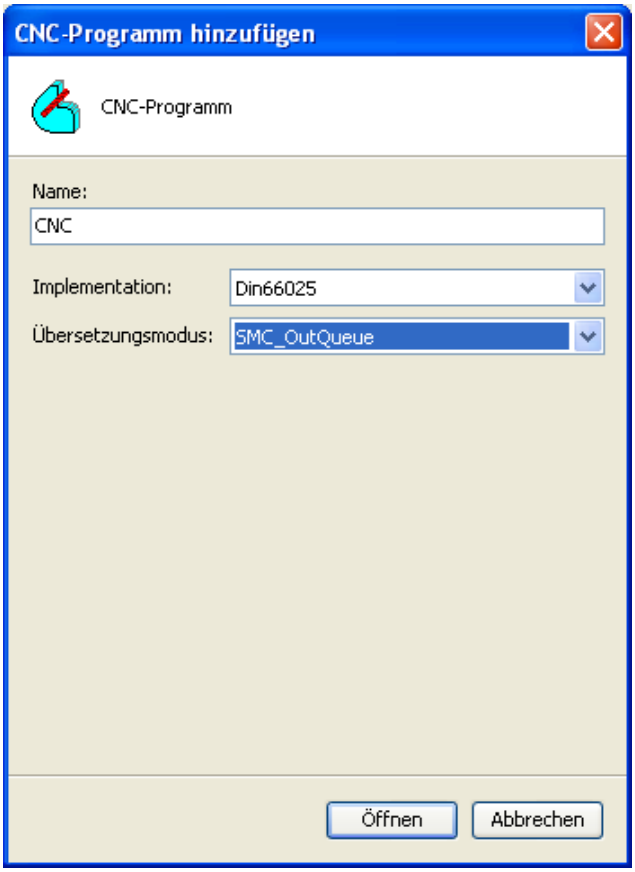


Abb. 316 "CNC Programm hinzufügen"- Dialog

Darin gibt man einen Namen, eine Implementation und den Übersetzungsmodus (standardmäßig "CNC", "Din66025" und "SMC_OutQueue") ein und bestätigt dies mit **Öffnen**. Das CNC-Objekt wird in den Gerätebaum eingefügt und der CNC-Editor öffnet mit der leeren Ansicht des neuen CNC-Objekts.

Editieren eines CNC-Objekts

Ein CNC-Objekt kann in vier Komponenten der Benutzeroberfläche angezeigt und bearbeitet werden, wobei die ersten drei automatisch beim Erzeugen eines CNC-Objektes öffnen: Haupteditor, Werkzeuge und Eigenschaften der Bahnelemente, außerdem Eigenschaften des CNC-Objekts.

Haupteditor

Der Haupteditor ist betitelt mit dem Namen des CNC-Programms und nachfolgend dem Pfad im Gerätebaum (z.B. CNC [Device: SPS- Logic: Application]). Das Fenster ist geteilt in einen oberen Teil, wo das DIN66025-Programm aufgelistet im Texteditor oder die CNC-Bahn in Tabellenansicht dargestellt wird. Dieser ist bei Start aktiv. Im unteren Teil des Fensters wird die grafische Ansicht der Bahn gezeigt. Mit Drücken von **F6** wechselt der Fokus zwischen diesen beiden Fenstern. Wird ein Bahnelement im grafischen Editor selektiert, dann ist implizit im Texteditor bzw. im Tabelleneditor der zugehörige Satz bzw. die zugehörige Zeile angewählt. Der Satz im Texteditor ist hellblau hervorgehoben. Die Zeile im Tabelleneditor ist hellgrau hervorgehoben. Umgekehrt wird ein im Tabellen- oder Texteditor angewähltes Elemente im grafischen Editor ebenfalls selektiert. Änderungen im grafischen Editor werden im Text- bzw. im Tabelleneditor übernommen und umgekehrt.

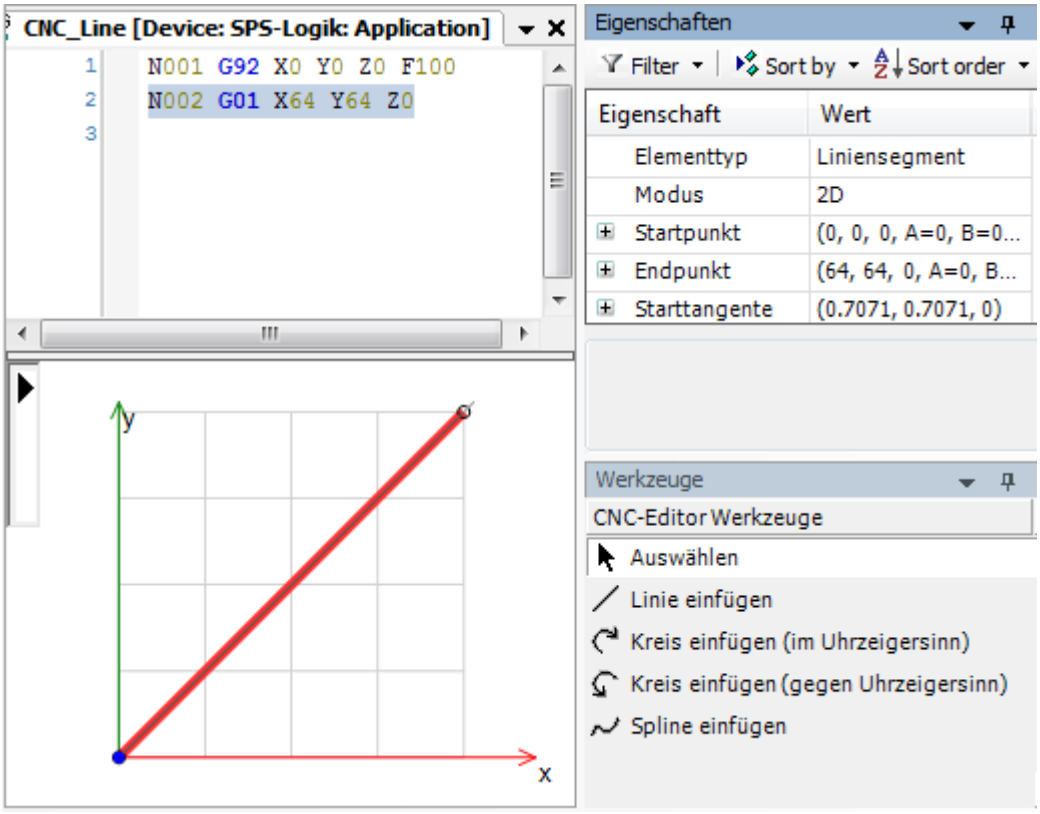


Abb. 317 CNC-Programm in DIN66250

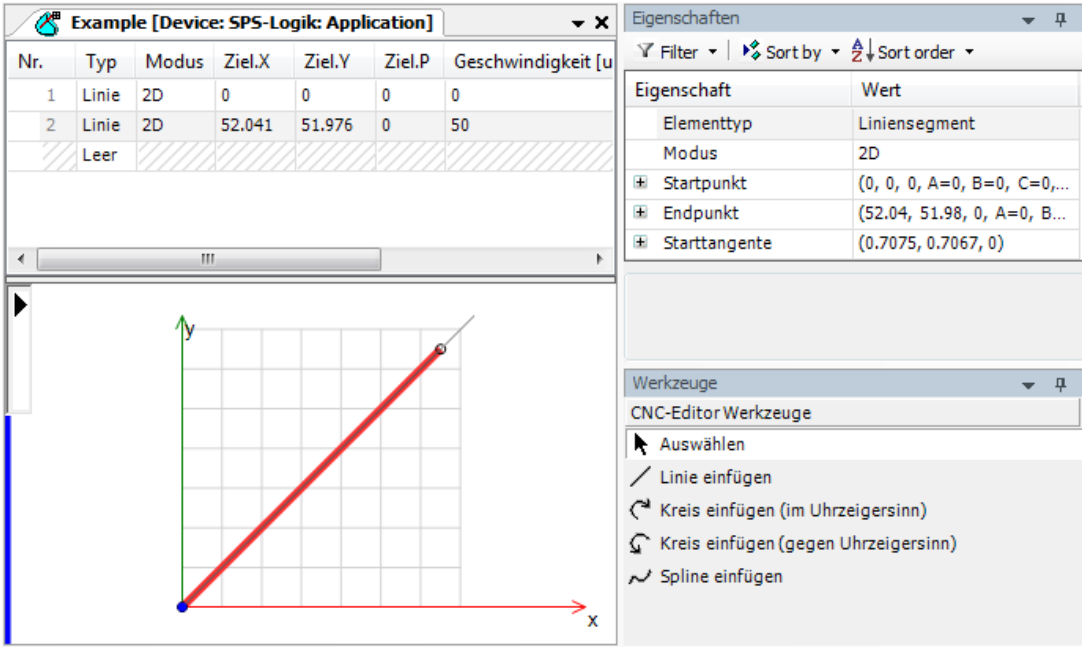


Abb. 318 CNC-Programm in Tabellenansicht

Ein CNC-Programm kann von Tabellenansicht zu Textansicht und umgekehrt transformiert werden.

Werkzeuge

Das Werkzeuge-Fenster stellt Werkzeuge zum Wählen und Einfügen von Bahnelementen zur Verfügung.

Eigenschaften-Fenster der Bahnelemente

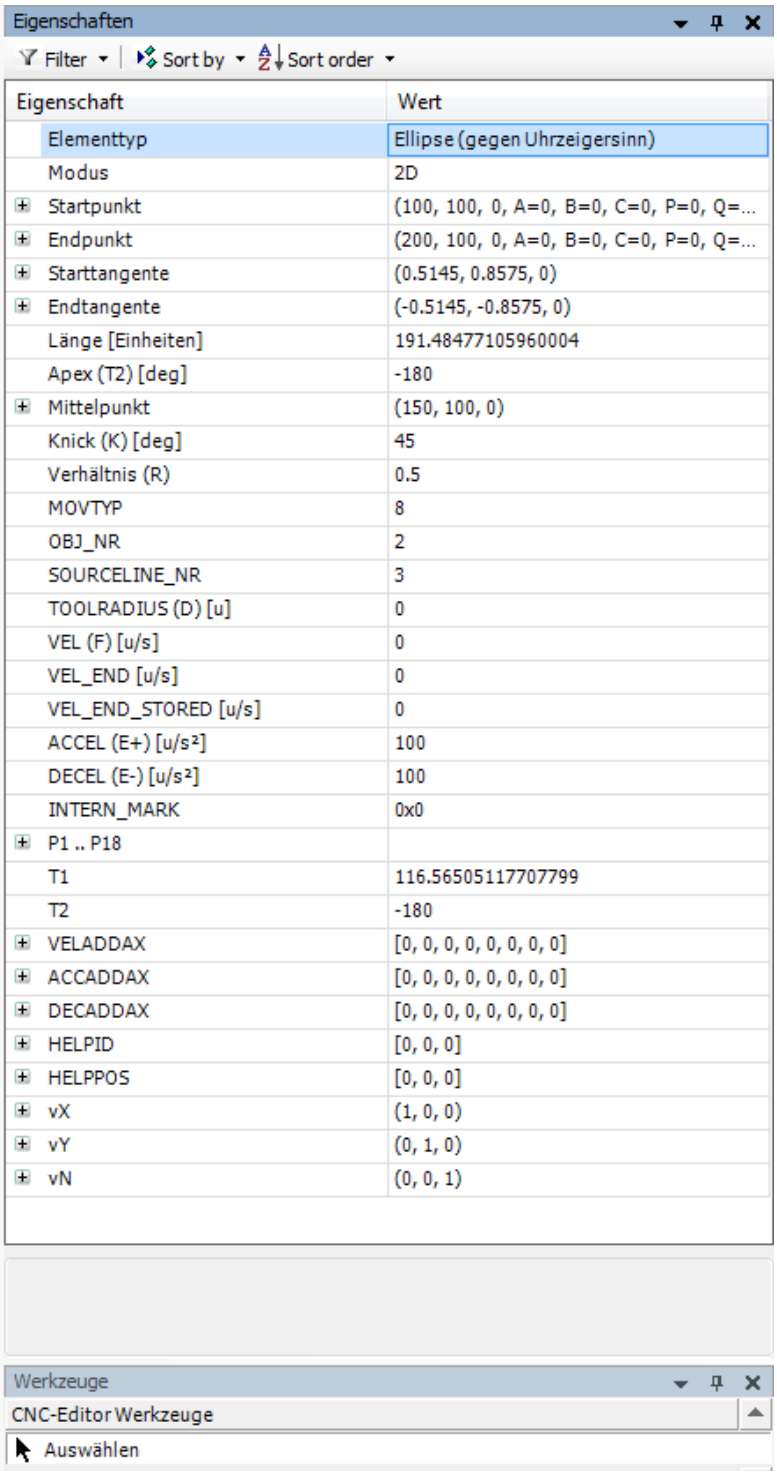


Abb. 319 CNC-Bahnelemente im Eigenschaften-Fenster

Die Eigenschaften der Bahnelemente werden im **Eigenschaften**-Fenster dargestellt, das standardmäßig beim Werkzeuge-Fenster verfügbar ist. Außerdem kann es mit einem Klick auf den Eintrag **Elementeigenschaften** im Menü **Ansicht** geöffnet werden. Dann werden alle Eigenschaften des selektierten Bahnelements aufgelistet.

Eigenschaften-Fenster des CNC Objekts

Der Eigenschaften-Editor des CNC-Objekts enthält allgemeine Informationen des CNC-Programms. Es wird entweder über den Eintrag **Eigenschaften...** im Kontextmenü des CNC-Objekts im Gerätebaum oder über den Eintrag **Eigenschaften** ( ) in der Kategorie **Ansicht** geöffnet.

Kompilieren des CNC-Programms

Beim Kompilieren werden Strukturvariablen vom Typ **SMC_OutQueue** bzw. **SMC_CNC_REF** (festgelegt in der Option Übersetzungsmodus) erzeugt, unabhängig davon, ob der Texteditor oder die Tabellenansicht zur Code-Generierung verwendet wurde. Diese Strukturen sind Teil der SM3_CNC Bibliothek. Die Datenstrukturen werden in einer globalen Variablen zusammengefasst und ermöglichen einen einfachen Zugriff, siehe CNC-Datenstrukturen.

Eine weitere Option ist es, eine Datei mit einer programmierten CNC Bahn während des Kompilierens zu erzeugen und beim Einloggen zur SPS zu übertragen. Dazu muss die Option Datei als Übersetzungsmodus ausgewählt werden.

Ein weitere Beschreibung finden Sie unter Übersetzungsmodus.

Öffnen eines existierenden Projekts

Um ein existierendes Projekt zu öffnen, geht ein Standard-Dialog auf. Eine detaillierte Beschreibung ist unter **Projekt öffnen...** zu finden. Handelt es sich um ein »PLC Designer« V2.3 Projekt, steht ein geeigneter Konverter zur Verfügung.

Folgende Aufgaben übernimmt der Konverter, falls ein SoftMotion fähiges Gerät zur Verfügung steht:

- Der Konverter wandelt jedes CNC-Programm in ein CNC-Objekt mit gleichem Namen und Implementationstyp **Din66025** um. Hier wird auch der Übersetzungsmodus, die Queue-Größe und die Startposition des konvertierenden Programms ebenso wie der Status der Satzunterdrückung gespeichert.
- Falls im CNC-Programm im V2.3 Projekt die Option **Vom Übersetzen ausschließen** aktiviert ist, ist die entsprechende Option **Vom Übersetzen ausschließen** in »PLC Designer« V3 ebenfalls gesetzt.
- Die in »PLC Designer« V2 verwendeten Werte Epsilon und Winkeltoleranz werden ignoriert.
- Ein CNC-Einstellungen-Objekt wird automatisch erzeugt.
- Alle CNC-Objekte und das CNC-Einstellungen-Objekt werden unterhalb einer SoftMotion fähigen Applikation eingeordnet.
- Steht kein SoftMotion fähiges Gerät zur Verfügung werden die CNC-Programme nicht konvertiert und eine Fehlermeldung ausgegeben.

13.3.1 Texteditor

CNC-Texteditor

Der Texteditor befindet sich im oberen Teil des CNC-Fensters. Hier kann ein CNC-Programm gemäß Din66025 eingegeben und geändert werden. Bahnelemente werden entsprechend im grafischen Editor angezeigt und können dort auch geändert werden. In diesem Fall wird der Textteil automatisch aktualisiert und umgekehrt.

Allgemein

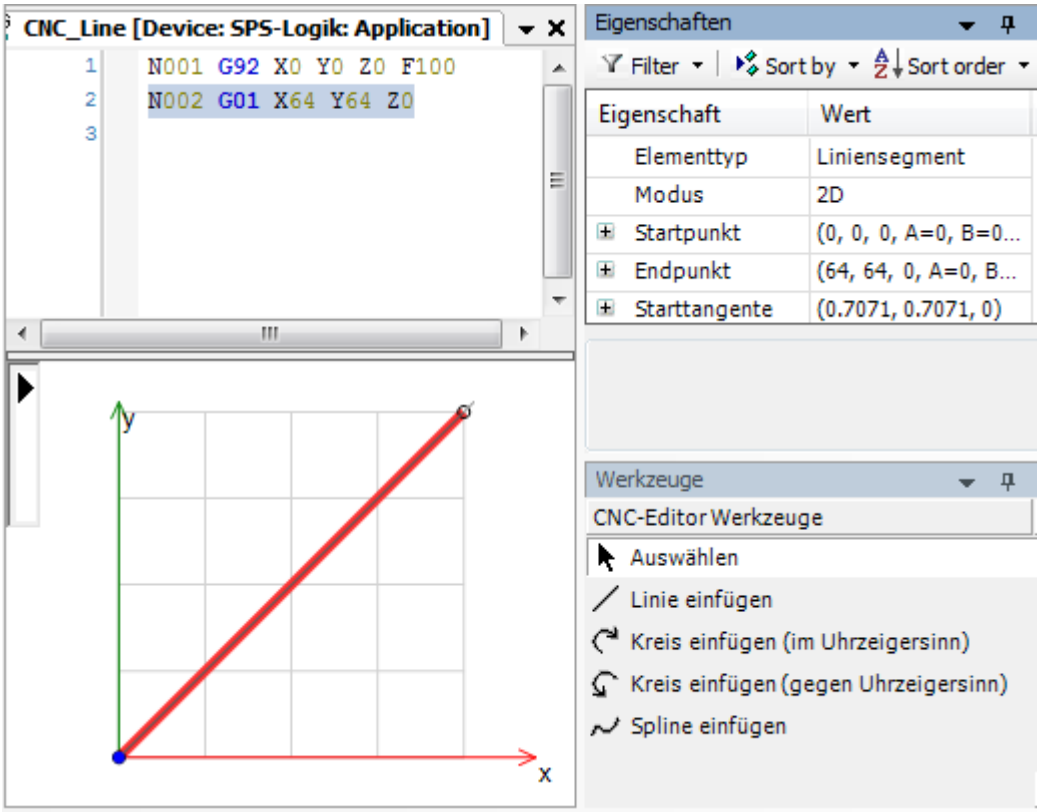


Abb. 320 CNC-Programm in DIN66250

Durch Klicken in den Texteditor wird der Satz hellgelb hervorgehoben und das zugehörige Bahnelement wird rot mit einem blauen Startpunkt und einem schwarz umrandeten Endpunkt dargestellt. Wird ein Bahnelement im grafischen Editor selektiert, wird der zugehörige Satz hellblau hervorgehoben. Mit Drücken von F6 wechselt der Fokus in den grafischen Editor und zurück.

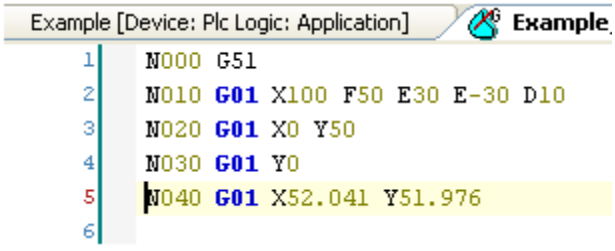


Abb. 321 Ansicht im Texteditor

Hinzufügen eines Satzes

Um ein Bahnelement einzufügen oder anzuhängen ist eine neue Zeile unter Verwendung der **Eingabetaste** zu erzeugen und das Bahnelement konform zu DIN66025 zu programmieren.

Ändern eines Satzes

Alle Sätze können durch direktes Editieren direkt im Texteditor geändert werden. Das geänderte Element ist dann wie gewohnt im grafischen Editor markiert. Durch Drücken von **F2** öffnet sich die Eingabehilfe und es können globale Variablen zum Einfügen in das CNC-Programm ausgewählt werden. Wurde als Übersetzungsmodus SMC_OutQueue ausgewählt, was dazu führt, dass »PLC Designer« eine fertige OutQueue-Struktur während des Kompilierens erzeugt, dann wird die Referenz der globalen Variable bereits während des Kompilierens ersetzt durch den Initialwert, was den Gebrauch von globalen Variablen in diesem Kontext sinnlos werden lässt.

Löschen eines Satzes

Ein selektierter Satz kann durch Verwenden der **Entf**-Taste gelöscht werden.

Befehle und Optionen

Verschiedene Befehle und Optionen stehen zur Verfügung, um zusätzliche Datensätze des CNC-Programms anzuzeigen oder zu manipulieren. Sie sind im CNC-Menü zu finden.

Hinweis: Bitte beachten, dass Referenzen von globalen Variablen im Decoder-Modul evaluiert werden, wenn der Decoder die Sätze abarbeitet. Dies kann einige Zyklen im Voraus geschehen, bevor das Objekt abgefahren wird.

Beispiele

Verschiedene CNC-Programme sind im Lieferumfang enthalten. Betrachten sie näher deren ausführliche Beschreibung.

Din66025

CNC-Sprache DIN66025

Um einen einfachen Weg des Programmierens von geometrischen Bahnen zu ermöglichen, unterstützt SoftMotion Teile der CNC-Sprache DIN66025. Da das ganze SoftMotion-Konzept in die viel mächtigere Sprache IEC61131 eingebettet ist, werden nur die Teile unterstützt, die zur Erstellung einer Bahn dienen.

Vorgeschriebene Struktur eines CNC-Programms:

```
N<Zeilennummer> G<Nummer>
```

Beispiel:

```
% MyExample
N10 G01 X100 Y100 E100 F100 E-200
N20 G01 Z40 F20
N30 G03 X-100 R200 F100
```

Syntax

Ein SoftMotion-CNC-Programm kann mit einem Prozent-Zeichen % beginnen. In der gleichen Zeile kann optional der Name des Programms getrennt durch ein Leerzeichen oder TAB stehen. Das eigentliche CNC-Programm setzt sich aus mehreren Sätzen zusammen.

Jeder Satz (Zeile) besteht aus beliebig vielen Worten.

Ein Wort besteht aus einem Buchstaben (Wortkennung) und einer nachfolgenden Zahl (z.B. G01; siehe auch untenstehende Liste). Groß- und Kleinschreibung sind ebenso belanglos wie führende Nullen (G01 = g1).

Das erste Wort jedes Satzes bildet die Satznummer (N<Zahl>) z.B. "N01". Die Worte eines Satzes sind durch **Leerzeichen** oder **Tabs** getrennt. Sie werden von rechts nach links abgearbeitet. Dabei bewirken alle Worte außer den Fahrbefehlen (G<Zahl>, z.B. "G02"), dass die Zahl des Wortes in eine Variable gemäß dem Buchstaben des Wortes geschrieben wird, auf welche schließlich der Fahrbefehl zugreift.

Jeder Satz darf nur einen Fahrbefehl enthalten, der direkt rechts neben der Satznummer stehen muss. Wird der Fahrbefehl in einem Satz nicht eingetragen, so wird automatisch der im letzten Satz verwendete Fahrbefehl ergänzt.

Jeder Fahrbefehl kann als Wegobjekt (Gerade, Kreisbogen) verstanden werden. Die Geschwindigkeit, mit welcher die Bahnobjekte interpoliert werden, richtet sich grundsätzlich nach der eingestellten Sollgeschwindigkeit, -beschleunigung und -verzögerung. Der Interpolator hat sicherzustellen, dass diese Grenzwerte nicht überschritten werden.

Die Geschwindigkeit während des Übergangs zweier benachbarter Objekte wird nach folgenden Regeln bestimmt:

- Eines der beiden Objekte ist eine Positionierung (G0). Übergangsgeschwindigkeit = 0.
- Der Winkel zwischen den Tangenten der beiden Objekte am Übergang ist größer als die Winkeltoleranz. Übergangsgeschwindigkeit = 0.
- Ansonsten gilt, dass die Übergangsgeschwindigkeit die kleinere Sollgeschwindigkeit der beiden Wegobjekt ist.
- Grundsätzlich bewirkt ein Fahrbefehl, dass eine Interpolation von der Zielposition des letzten Fahrbefehls zur Zielposition, die vom aktuellen Fahrbefehl spezifiziert wurde, durchgeführt wird. Der erste Fahrbefehl startet bei der spezifizierten Position (festgelegt im Decoder oder CNC-Editor). Falls diese Position nicht definiert wurde, wird an der Position X=0, Y=0, Z=0 gestartet.

Zusätzlich besteht die Möglichkeit, mittels G92 die Position im CNC-Programm zu setzen. Dies ist sowohl am Anfang des CNC-Programms möglich (dort bewirkt es das Setzen der Startposition) als auch in der Mitte des Programms, wo es einen Sprung der Soll-Position auf die mit G92 definierte Position zur Folge hat. Bei mehreren aufeinanderfolgenden G92-Befehlen überwiegt der letzte; die vorherigen werden also übersprungen. Will man aber sicherstellen, dass auch die vorigen G92-Positionen (für einen Zyklus lang) ausgegeben werden, fügt man dazwischen den Befehl G1 mit identischen Koordinaten ein. Dieses Mittel setzt man ein, wenn die Bahn zwischen diesen Positionen nicht von Interesse ist, sondern die Sollposition auf schnellstmöglichem Weg dorthin gelangen soll. Die Bausteine SMC_ControlAxisByPos detektieren dann einen Sprung in den Sollwerten, halten den Interpolator an und interpolieren jede Achse einzeln auf schnellstmöglichem Weg dorthin.

Beispiel:

```
%MyExample
N10 G92 X100 Y100 (Sollposition auf 100/100 setzen)
N20 G1 X100 Y100 (Einmalige Ausgabe der Position sicherstellen)
N30 G92 X50 Y100 (Sollposition auf 50/100 setzen)
```

Ein mit dem Zeichen **/** beginnender Satz wird bei eingeschalteter Satzunterdrückung im CNC-Menü übersprungen.

Zeichen, die innerhalb zweier runder Klammern stehen, werden als Kommentare aufgefasst und haben keinen Einfluss auf die programmierte Bahn. Verschachtelte Kommentare werden nicht unterstützt.

Die Satznummer (N<Zahl>) wird als Marker und für die Festlegung von Sprungzielen verwendet, hat aber sonst keine weitere Bedeutung.

Alle Zahlenwerte außer die der Fahrbefehle (G<Zahl>), der M-Funktion (M<Nummer>) und der Schaltpunktnummer (H<Zahl>) können Fließkommawerte sein.

Siehe dazu:

- Positionierung
- Gerade
- Kreisbogen
- Verzugszeit
- Spline
- Parabel
- Ellipse
- 3D-Modus
- Sprünge
- Ändern von Variablenwerten
- Vorverarbeitung
- Koordinatenverschiebung
- Zeitsynchronisation mit Interpolation
- Modi
- Schaltpunkt (H-Funktion)
- M-Funktion
- Verwendung von Variablen
- Zusatzfunktion (M-Funktion)

Wortkennung :

Buchstabe	Funktion
A	Zielposition für zusätzliche Spline-Achse A.
B	Zielposition für zusätzliche Spline-Achse B.
C	Zielposition für zusätzliche Spline-Achse C.
D	Werkzeugradius (für Korrektur G40-42 bzw. Eckverrundung G50-51) bzw. Variablenwert (G36/G37)
E	Max. Beschleunigung (>0) / Verzögerung (<0) [Wegeinheiten/sec^2]
F(feed)	Geschwindigkeit, mit der gefahren werden soll [Wegeinheiten/sec]
G (geometry)	Fahrbefehl
H	Schaltpunkt einschalten (>0) / ausschalten (<0)
I	X-Koordinate vom Kreis-/Ellipsenmittelpunkt (G02/G03/G08/G09) oder X-Koordinate vom Parabel-Tangentenschnittpunkt

Buchstabe	Funktion
J	Y-Koordinate vom Kreis-/Ellipsenmittelpunkt (G02/G03/G08/G09) oder Y-Koordinate vom Parabel-Tangentenschnittpunkt
K	Richtung der Ellipsenhauptachse im mathematischen Sinn (0° O, 90° N,...) bzw. Sprungbedingung (G20) bzw. dT1-Parameterwert (M-Funktion) oder Z-Koordinate des Kreismittelpunktes (G02/G03 im 3D-Modus)
L	absolute Schaltpunktposition gemessen vom Wegobjektanfang (>0) bzw. Wegobjektende (<0) bzw. Sprungziel (G20) bzw. dT2-Parameterwert (M-Funktion)
M (machine)	Zusatzfunktion (M-Funktion)
O	relative Schaltpunktposition [0..1] bzw. zu verändernde Variable (G36/G37) oder M-Parameter-Datenstruktur (M)
P	Zielposition der zusätzlichen linearen Achse P.
Q	Zielposition der zusätzlichen linearen Achse Q.
R	Kreisradius (G02/G03) (alternativ zu I,J) oder Längenverhältnis der elliptischen Nebenachse/Hauptachse (G08/G09)]0..1]
S	S-Profil für lineare Achsen ein-(>0)/ausschalten(<0) 3: Z-Achse (nur 2D-Modus) , 7: P-Achse, 8: Q-Achse, 9: U-Achse, 10: V-Achse, 11: W-Achse
T	Kommando spezifischer Parameter
U	Zielposition der zusätzlichen linearen Achse U
V	Zielposition der zusätzlichen linearen Achse V
W	Zielposition der zusätzlichen linearen Achse W
X	X-Koordinate der Zielposition
Y	Y-Koordinate der Zielposition
Z	Z-Koordinate der Zielposition

Kommandos:

	Funktion
G00	Direkte Bewegung ohne Werkzeugeingriff, Positionierung
G01	Lineare (gerade) Bewegung mit Werkzeugeingriff
G02	Kreis(-stück) im Uhrzeigersinn
G03	Kreis(-stück) gegen den Uhrzeigersinn
G04	Verzögerungszeit
G05	Punkt einer 2D Kardinalspline

	Funktion
G06	Parabel
G08	Ellipse(-nsegment) im Uhrzeigersinn
G09	Ellipse(-nsegment) gegen den Uhrzeigersinn
G10	Punkt der 3D Kardinalspline
G15	Wechsel zu 2D
G16	Wechsel zu 3D durch Aktivierung des 3D Modus mit Normalenvektor I/J/K zur Ebene.
G17	Wechsel zu 3D durch Aktivierung des 3D-Modus in X/Y-Ebene.
G18	Wechsel zu 3D durch Aktivierung des 3D-Modus in Z/X-Ebene.
G19	Wechsel zu 3D durch Aktivierung des 3D-Modus in Y/Z-Ebene.
G20	Bedingter Sprung (nach L, wenn K <> 0)
G36	Wert (D) auf Variable (O) schreiben.
G37	Variable (O) um Wert (D) inkrementieren.
G40	Ende der Werkzeugradiuskorrektur.
G41	Beginn der Werkzeugradiuskorrektur links der Fahrrichtung
G42	Beginn der Werkzeugradius-Korrektur rechts der Fahrrichtung
G50	Ende der Eckverrundung/-schleifung
G51	Beginn der Eckverschleifung
G52	Beginn der Eckverrundung
G53	Ende der Funktion zur Koordinatenverschiebung.
G54	Setzen des Offsets aller folgenden Koordinaten auf eine spezifizierte Position.
G55	Addieren der spezifizierten Position zur aktuellen Position.
G56	Setzen des Offsets so, dass die aktuelle Position gleich der spezifizierten Position ist.
G60	Ende der Avoid-Loop-Funktion (Schleifenvermeidung)
G61	Beginn der Avoid-Loop-Funktion (Schleifenvermeidung)
G75	Zeitsynchronisation mit dem Interpolator.
G90	Beginn der Interpretation der folgenden Koordinatenwerte (für X/Y/Z/P-W/A/B/C) als absolute Werte (standardmäßig)
G91	Beginn der Interpretation der folgenden Koordinatenwerte (für r X/Y/Z/P-W/A/B/C) als relative Werte.
G92	Setzen der Position ohne Bewegung.
G98	Beginn der Interpretation der folgenden Koordinatenwerte von I/J/K als absolute Werte.
G99	Beginn der Interpretation der folgenden Koordinatenwerte von I/J/K als Werte relativ zur Startposition (standardmäßig)

Man beachte, dass die Bibliothek SM3_CNC eingebunden werden muss, um ein fehlerfreies Übersetzen des Projekts zu ermöglichen.

Positionierung

Kommando G0, G92

Dieses Kommando wird verwendet um Bahnobjekte in DIN66025 zu positionieren.

Zulässige Syntax

G0 X Y Z A B C P Q U V W F E H L/O S

G92 X Y Z A B C P Q U V W H L/O

Parameter der Positionierungsbefehle	
X Y Z	Zielpositionen der kartesichen Achsen
A B C P Q U V W	Zielpositionen der Zusatzachsen
F E	Bahngeschwindigkeit, -beschleunigung, -verzögerung G-Code Wortkennung (Buchstabenadresse)
H L/O	Schaltpunkt (H-Funktion)
S	S-Profil G-Code Wortkennung (Buchstabenadresse)

Grafische Darstellung

Im grafischen Editor werden G0-Befehle mit grünen Linien dargestellt. G92 Befehle werden nicht direkt dargestellt, sie verschieben lediglich den Anfangspunkt des folgenden Elements. Positionierungsbefehle können nur im CNC-Texteditor oder im Tabelleneditor eingefügt werden. Im grafischen Editor können sie verschoben werden.

Beispiel

Positionierung mit Geradeninterpolation mit definiertem Vorschub

G-Code

```
N0 G92 X10 Y10 (Setzen der Position auf 10/10)
N10 G1 X20 F10 (Geraden-Interpolation nach 20/10 mit Vorschub 10)
N20 G0 Y20 F100 (Geraden-Positionierung nach 20/20 mit Vorschub 100)
```

Beispiel

Verschiedene Positioniermodi

G-Code

```
N0 G92 X100 Y100 F100 (Setzen der Position auf 100/100)
N10 G1 X100 Y100 (Sichert die Ausgabe dieser Position für einen Zyklus)
N20 G92 X50 Y100 (Setzen der Position sprunghaft auf 50/100)
```

Gerade

Kommando G1

Die Geraden-Interpolation bewirkt, dass sich der Interpolationspunkt von der aktuellen

Position zu den definierten Zielkoordinaten auf einer Gerade bewegt. Alle Achsen kommen gleichzeitig dort an. Für die Beschreibung der Geraden müssen die Zielkoordinaten (X/Y/Z) definiert werden.

Zusätzlich zur Bewegung im Raum können für die Zusatzachsen A/B/C/P/Q/U/V/W Zielpositionen angegeben werden, die gleichzeitig mit den kartesischen Zielkoordinate erreicht werden.

Das Geradenelement kann mit Hilfe der „Werkzeuge“ eingefügt werden.

Zulässige Syntax

G1 X Y Z A B C P Q U V W F E H L/O D S

Parameter des Positionierungsbefehls	
X Y Z	Zielpositionen der kartesichen Achsen
A B C P Q U V W	Zielpositionen der Zusatzachsen
F E	Bahngeschwindigkeit, -beschleunigung, -verzögerung G-Code Wortkennung (Buchstabenadresse)
H L/O	Schaltpunkt (H-Funktion)
D	Werkzeugradius
S	S-Profil G-Code Wortkennung (Buchstabenadresse)

Grafische Darstellung

Im graphischen Editor wird eine Gerade mittels einer schwarzen Linie mit kleinen schwarzen Kreisen für Start- und Endposition dargestellt.

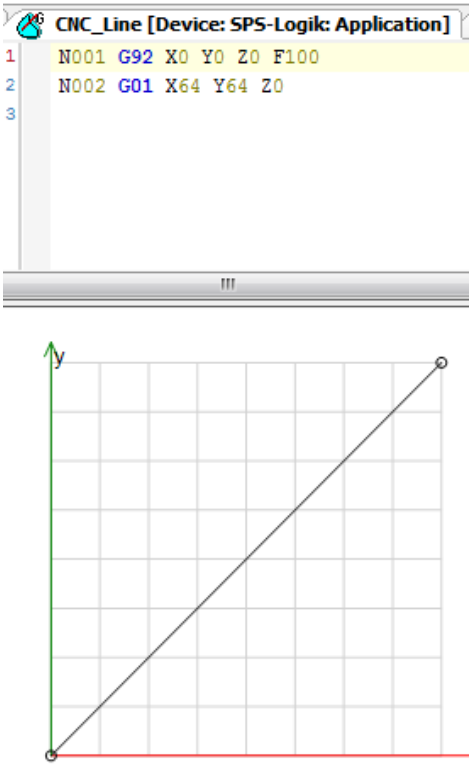


Abb. 322 CNC Element Gerade

Beispiel

Gerade von der Startposition [default 0/0/0] nach 1/2/3 mit Geschwindigkeit 1, Beschleunigung 10, Verzögerung 20. Gleichzeitig wird die Orientierungsachse A auf 180 Grad gefahren.

G-Code

```
N010 G01 X1 Y2 Z3 A180 F1 E10 E-20
```

Beispiel

Startposition setzen auf 50/50/50. Geradeninterpolation von 50/50/50 nach 64/30/0 mit Vorschub 100

G-Code

```
N000 G92 X50 Y50 Z50 F100 (Startposition setzen auf 50/50/50)
N010 G01 X64 Y30 Z0 (Geradeninterpolation von 50/50/50 nach 64/30/0 mit Vorschub 100)
```

Kreisbogen

Kommando G2, G3

Die Kreis-Interpolation bewirkt, dass sich der Interpolationspunkt von der aktuellen Position entlang einer Kreisbahn zur Zielposition bewegt. Das Kommando G2 bewirkt eine Bewegung im, das Kommando G3 gegen den Uhrzeigersinn.

Für die Beschreibung eines Kreises in DIN66025 müssen Zielkoordinaten (X/Y/Z) definiert werden. Die Biegung kann auf zwei Arten definiert werden: Entweder man gibt den Radius (R) des Kreises oder die Koordinaten des Mittelpunktes (I/J/K) ein.

Zusätzlich besteht die Möglichkeit, den Kreisbogen über Mittelpunktskoordinaten und Öffnungswinkel zu definieren; die Zielposition ergibt sich dann automatisch.

Man beachte, dass nicht ausführbare Vorgaben (z. B. Radius kleiner als halbe Distanz zwischen Start und Ziel oder definierter Mittelpunkt hat zu Start- und Zielposition unterschiedliche Distanz) bewirken, dass eine Gerade anstatt des Kreises gefahren wird.

Zulässige Syntax

G2 X Y Z R A B C P Q U V W F E H L/O D S
G2 X Y Z I J K A B C P Q U V W F E H L/O D S
G2 I J K T A B C P Q U V W F E H L/O D S
G3 X Y Z R A B C P Q U V W F E H L/O D S
G3 X Y Z I J K A B C P Q U V W F E H L/O D S
G3 I J K T A B C P Q U V W F E H L/O D S

Parameter	
X Y Z	Zielpositionen der kartesichen Achsen
R	Kreisradius
I J K	Kreismittelpunktskoordinaten
T	Öffnungswinkel in Grad
A B C P Q U V W	Zielpositionen der Zusatzachsen
F E	Bahngeschwindigkeit, -beschleunigung, -verzögerung
H L/O	Schaltpunkte
D	Werkzeugradius
S	S-Profil

Grafische Darstellung

Im grafischen Editor werden Kreisbögen in schwarz mit einem schwarzen Kreis für Start- und Endpunkt dargestellt. Ist das Bahnelement selektiert, wird es rot mit einem blauen Startpunkt und einem schwarzen Endpunkt visualisiert. Zusätzlich wird ein hellgrauer Kreis in der Mitte des Bogens dargestellt, wenn der Radius veränderbar ist. Es ist möglich, das Bahnelement durch Klicken auf die Kreise und Verschieben mit gedrückter rechter Maustaste in eine gewünschte Position zu ändern.

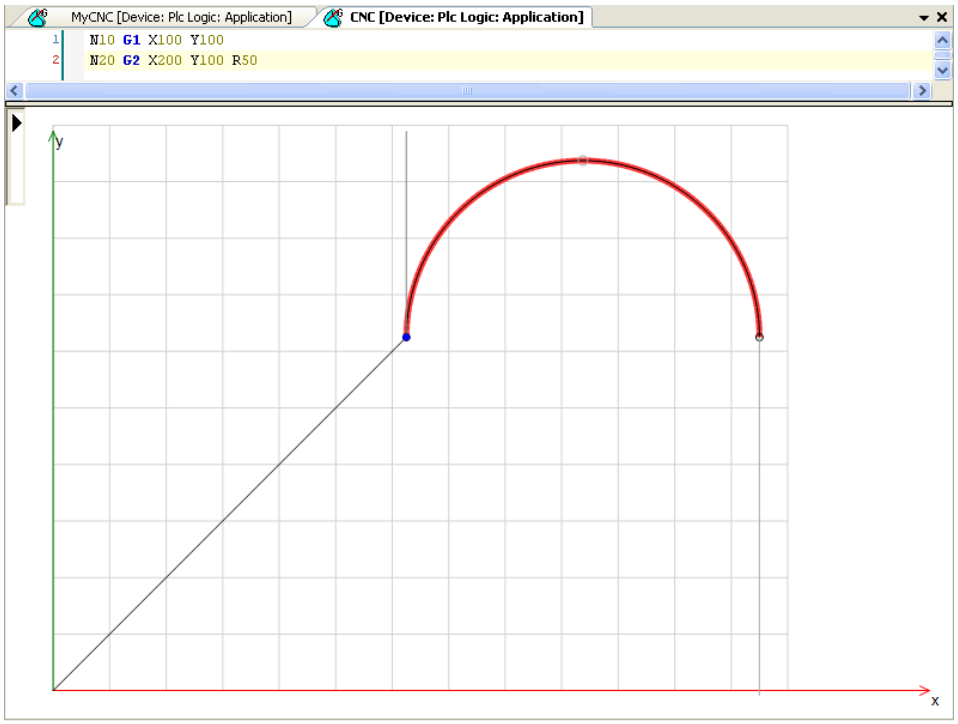


Abb. 323 Kreisbogen im grafischen Editor

Radius-Methode

Wird der Radius verwendet, ist zu beachten, dass nur Kreisbögen < 180° (Halbkreise) möglich sind, da die Radiusmethode (außer für den Fall des Halbkreises) immer zwei mögliche Lösungen beschreibt: Eine kleiner als ein Halbkreis, eine größer als ein Halbkreis. Das System verwendet immer die, die kleiner als ein Halbkreis ist.

Beispiel für einen Halbkreise:

```
N10 G1 X100 Y100
N20 G2 X200 Y100 R50
```

Will man einen Kreisbogen mit mehr als 180° Öffnungswinkel, so muss man ihn über (I/J/K) festlegen. Diese Methode ist eindeutig, ausgenommen in dem Fall, in dem Start-und Endpunkt des Kreises identisch sind. Dort könnte entweder ein Null- oder ein Vollkreis entstehen. Das System fügt in diesem Fall einen Vollkreis ein.

Mittelpunkt-Methode

Man beachte auch, dass bei der Methode (I/J/K) entscheidend ist, ob die Koordinaten I/J/K (siehe G98/G99) relativ oder absolut angegeben werden. Werden I, J und K nicht korrekt angegeben (Abstand zwischen Mittelpunkt und Start- bzw. Endpunkt muss identisch sein), dann ist kein Kreisbogen möglich und das System ersetzt diesen durch eine Gerade.

Beispiel für den gleichen Halbkreis (wie oben gezeigt) über relativen Mittelpunkt spezifiziert:

G-Code

```
N10 G1 X100 Y100 F100
N15 G99
N20 G2 X200 Y100 I50 J0
```

Beispiel für den gleichen Halbkreis über absoluten Mittelpunkt spezifiziert:

```
N10 G1 X100 Y100
N15 G98
N20 G2 X200 Y100 I150 J100
```

Öffnungswinkel-Methode

Will man den Kreis über den Öffnungswinkel T definieren, beachte man, dass maximal 3 Vollkreise zulässig sind (T>=1080).

Beispiel für den gleichen Halbkreis über Öffnungswinkel spezifiziert:

G-Code

```
N10 G1 X100 Y100
N15 G99
N20 G2 I50 J0 T180
```

Beispiel für eine Helix (2 volle Kreise)

G-Code

```
N10 G1 X100 Y100
N15 G99
N20 G2 I50 J0 T720
```

Kreisbögen im 3D-Modus

Im 3D-Modus legt das System den Kreisbogen durch die mittels der Kommandos G15-G19 definierte Interpolationsebene: „CNC-Sprache DIN 66025“.

Beispiel für einen Halbkreis in der X/Z-Ebene und einen Halbkreis in der X/Y-Ebene

G-Code

```
N10 G1 X100 Y100
N15 G18
N20 G2 X200 R50
N25 G17
N30 G3 X100 R50
```

Verweilzeit

Kommando G4

Mit diesem Kommando wird in DIN66025 eine Verweilzeit implementiert. Der Interpolator bleibt für eine gewisse Zeit an seiner aktuellen Position stehen.

Beispiel:

```
N0 G4 T1 (Verweilen von 1s)
```

Spline

Kommando G5, G10

Zur Beschreibung von Splines in DIN66025 genügt es, die Zielkoordinaten des nächsten Spline-Abschnitts einzugeben. Der Spline-Abschnitt wird vom System automatisch so berechnet, dass die Endtangente des vorigen Elements und die Starttangente des Splines, sowie die Endtangente des Splines und die Starttangente des folgenden Elements übereinstimmen, so dass kein Knick in der Bahn entsteht.

Man unterscheidet 2,5D-Splines (G5) und 3D-Splines (G10). Erstere bestehen aus Polynomialfunktionen in X und Y, sowie einer linearen Funktion in Z, während die zweite Variante auch in Z aus einer Polynomialfunktion berechnet werden.

Zulässige Syntax

G5 X Y Z A B C P Q U V W F E H L/O D S

G10 X Y Z A B C P Q U V W F E H L/O D S

Parameter der Befehle	
X Y Z	Zielpositionen der kartesichen Achsen
A B C P Q U V W	Zielpositionen der Zusatzachsen
F E	Bahngeschwindigkeit, -beschleunigung, -verzögerung G-Code Wortkennung (Buchstabenadresse)
H L/O	Schaltpunkte
D	Werkzeugradius
S	S-Profil G-Code Wortkennung (Buchstabenadresse)

Man beachte, dass die Spline-Interpolation rechenintensiver ist als die Interpolation einer Geraden oder eines Kreises, was gerade auf Steuerungen mit begrenzter Performance dazu führt, dass der Interpolationszyklus hoch gesetzt werden muss.

Beispiel

Treppenprofil mit Splines verrundet

```
N0 G0 X0 Y0 Z0 F100 (Startposition)
N10 G5 X20 Y0
N20 G5 X20 Y20
N30 G5 X40 Y20
N40 G5 X40 Y40
```

Abb. 324 Treppenprofil mit Splines verrundet

Es illustriert auch, wie das System die Start- und Endtangenten der einzelnen Spline-Abschnitte bestimmt: Am Splineanfang hängt die Starttangente davon ab, ob vorab schon Elemente mit Werkzeugeingriff vorhanden sind (G1, G2, G3, G8, G9) oder nicht (z.B. G0, G92, M). Im ersten Fall wird deren Endtangente als Start-Tangente für den Spline verwendet; im zweiten Fall wird die Verbindungslinie zwischen Startpunkt und erstem Splinepunkt als Tangente herangezogen. In der Mitte des Splines werden die Nachbarpunkte verbunden. Die Tangente des Punktes ist dann parallel zu dieser Verbindungslinie (s. grüne Gerade). Am Ende des Splines wird analog zum Splineanfang verfahren: Folgt ein Element mit Werkzeugeingriff, wird dessen Starttangente verwendet; wenn nicht, dient die Linie zwischen letztem und vorletztem Punkt.

Beispiel

Profil verrundet mit doppelter Spline:

```
N0 G0 X0 Y0 F100 (Startposition)
N5 G1 X5 Y0
N10 G5 X20 Y0
N20 G5 X20 Y20
N21 G5 X20 Y20
N30 G5 X40 Y20
N40 G5 X40 Y40
N45 G1 X0 Y40
```

Dieses Beispiel unterscheidet sich von dem vorigen dadurch, dass der Spline-Punkt bei X20 Y20 doppelt vorhanden ist. Dadurch wird die Spline unterbrochen und neu gestartet. Dies zeigt sich an der Festlegungsmethode der Tangente an diesem Punkt. Ein weiterer Unterschied zum vorangehenden Beispiel sind Start und Ende der Splines. Hier werden die Starttangenten durch das vorangehende bzw. nachfolgende Geradenstück festgelegt:

Parabel

Kommando G6

Für die Beschreibung einer Parabel in DIN66025 werden die Koordinaten des Endpunkts (X/Y/Z) und des Scheitelpunktes (I/J) festgelegt. Der Scheitelpunkt ist der Punkt, an dem die Tangente des Elements parallel zur Verbindungslinie von Start- und Endpunkt ist.

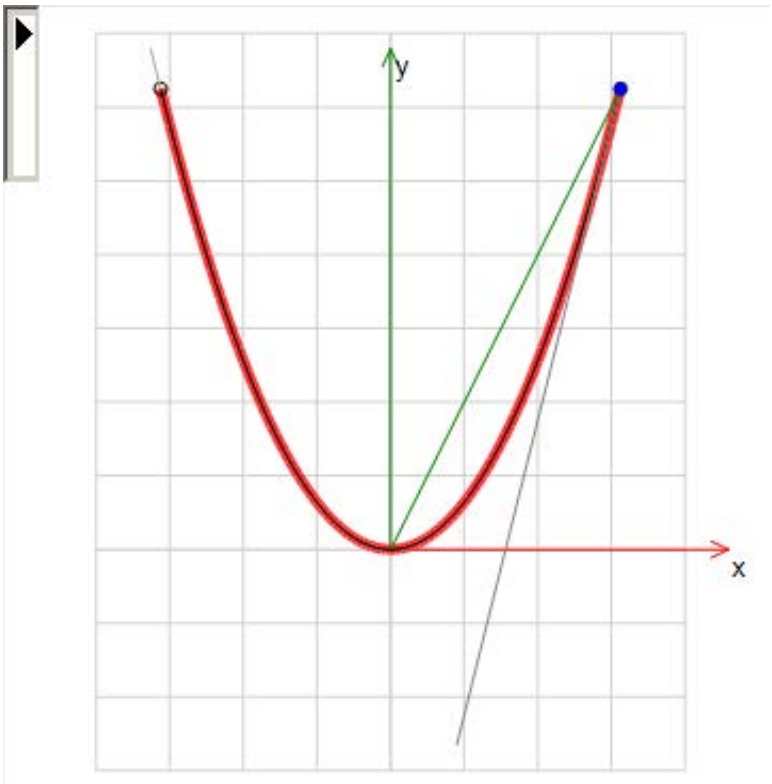


Abb. 325 CNC Parabelelement

Zulässiger Syntax

G6 X Y Z I J A B C P Q U V W F E H L/O D S

Parameter des Befehls

Parameter des Befehls	
X Y Z	Zielpositionen der kartesichen Achsen
I J	Scheitelpunkt der Parabel
A B C P Q U V W	Zielpositionen der Zusatzachsen
F E	Bahngeschwindigkeit, -beschleunigung, -verzögerung G-Code Wortkennung (Buchstabenadresse)
H L/O	Schaltpunkt (H-Funktion)
D	Werkzeugradius
S	S-Profil G-Code Wortkennung (Buchstabenadresse)

Hinweis: Man beachte, dass es sich bei diesem Objekt um ein 2,5-D-Objekt handelt, d. h. die Parabel liegt immer in der X/Y-Ebene. Parabeln in anderen Raumebenen werden nicht unterstützt

Beispiel

G-Code

```
N010 G00 X100 Y200 Z0 F100
N020 G98
N030 G06 X-100 Y200 Z00 I0 J0
```

Ellipse

Kommando G8, G9

Zur Beschreibung eines Ellipsenbogens in DIN66025 müssen die Zielkoordinaten (X/Y), der Ellipsenmittelpunkt (I/J), die Ellipsenhauptachsen-Richtung (K) sowie das Längenverhältnis zwischen Haupt- und Nebenachse (R) spezifiziert werden.

Erlaubte Syntax

G8 X Y Z R I J K A B C P Q U V W F E H L/O D S

G9 X Y Z R I J K A B C P Q U V W F E H L/O D S

Parameter	
X Y Z	Zielpositionen der kartesichen Achsen
R	Kreisradius
I J	Mittelounkt
K	Richtung der Ellipsenhauptachse in Grad (0: in Richtung der X-Achse, 90: in Richtung der Y-Achse, -90: entgegengesetzt zur Y-Achse)
A B C P Q U V W	Zielpositionen der Zusatzachsen
F E	Bahngeschwindigkeit, -beschleunigung, -verzögerung
H L/O	Schaltpunkte
D	Werkzeugradius
S	S-Profil

Das Längenverhältnis von Haupt- zur Nebenachse (R) muss im Bereich [0, 1] liegen. Es wird nur dann verwendet, wenn der Ellipsenbogen durch die Endpunkte, den Mittelpunkt und die Neigung der Hauptachse nicht eindeutig definiert ist. Dies ist der Fall, wenn beide Endpunkte denselben Abstand zur Hauptachse haben. (Dann müssen auch beide Endpunkte denselben Abstand zur Nebenachse haben, sonst gibt es keine Ellipse durch diese Punkte und das System ersetzt die Ellipse durch eine Gerade.)

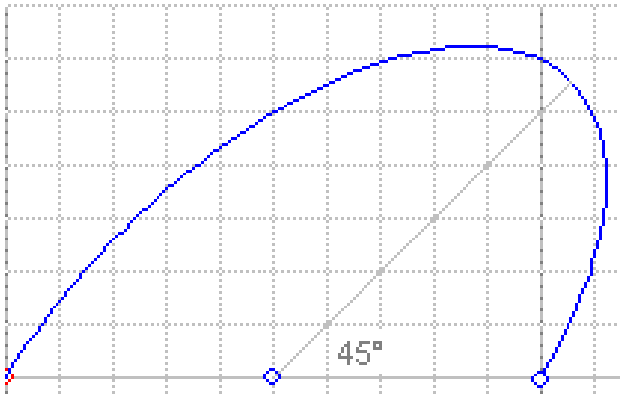


Abb. 326 Ellipse

Hinweis: Man beachte, dass es sich bei diesem Objekt um ein 2,5-D-Objekt handelt, d.h. die Parabel liegt immer in der X/Y-Ebene. Parabeln in anderen Raumebenen werden nicht unterstützt.

Beispiel:

G-Code

```
N10 G0 X100 Y100
N15 G98
N20 G8 X200 Y100 I150 J100 K45 R0.5
```

3D-Modus

In diesem Modus der DIN66025 hat der Anwender die Möglichkeit, dreidimensionale Bahnelemente zu verwenden. So können mehrere Punkte im 3D-Raum gesetzt und mit 3D-Splines verbunden, oder Kreisbögen in einer beliebigen Raumebene eingefügt werden. Im Unterschied zu 2.5-dimensionalen Bahnelementen, die nur X/Y-Koordinaten verwenden, bezieht sich die Bahngeschwindigkeit und Beschleunigung auf die Bewegung in den Koordinaten X/Y/Z. (Die Z-Achse wird gleich behandelt wie die X- und Y-Achse und so kann jede Bewegung in Z-Richtung implementiert werden.)

Hauptunterschiede zwischen 2.5D und 3D

- Die Definition der Geschwindigkeit und Beschleunigung (F, E) wird auf verschiedene Weise angewendet:
  - In 3D: Hier wird es für die Geschwindigkeit/Beschleunigung der Bahn in 3D verwendet.
  - In 2.5D: Hier wird es für die Geschwindigkeit/Beschleunigung der Projektion der Bahn auf die X/Y-Ebene verwendet.
- Splines in 3D werden im 3D-Modus (G10) mit einer ruckfreien Z-Komponente unterstützt. Die Splines in 2.5D sind in Z-Richtung nicht geglättet.
- Kreisbögen können in 3D in jeder möglichen Ebene im Raum definiert werden. In 2.5 ist dies begrenzt auf die X/Y-Ebene.

Beispiel:

```
N0G1 X1 Z100 F1
```

In 3D benötigt diese Bewegung ungefähr 100 Sekunden, da die Länge des Objekts die Z-Komponente enthält.

Länge = $(1^2 + 100^2)^{1/2}$

Bei einer Geschwindigkeit von 1 werden ungefähr 100 Sekunden benötigt. Hier ist die Geschwindigkeit der Z-Komponente ungefähr 1.

In 2.5D benötigt diese Bewegung ungefähr 1 Sekunde für die Kalkulation der Länge = 1. Hier ist die Geschwindigkeit in Z-Richtung ungefähr 100.

Allgemein

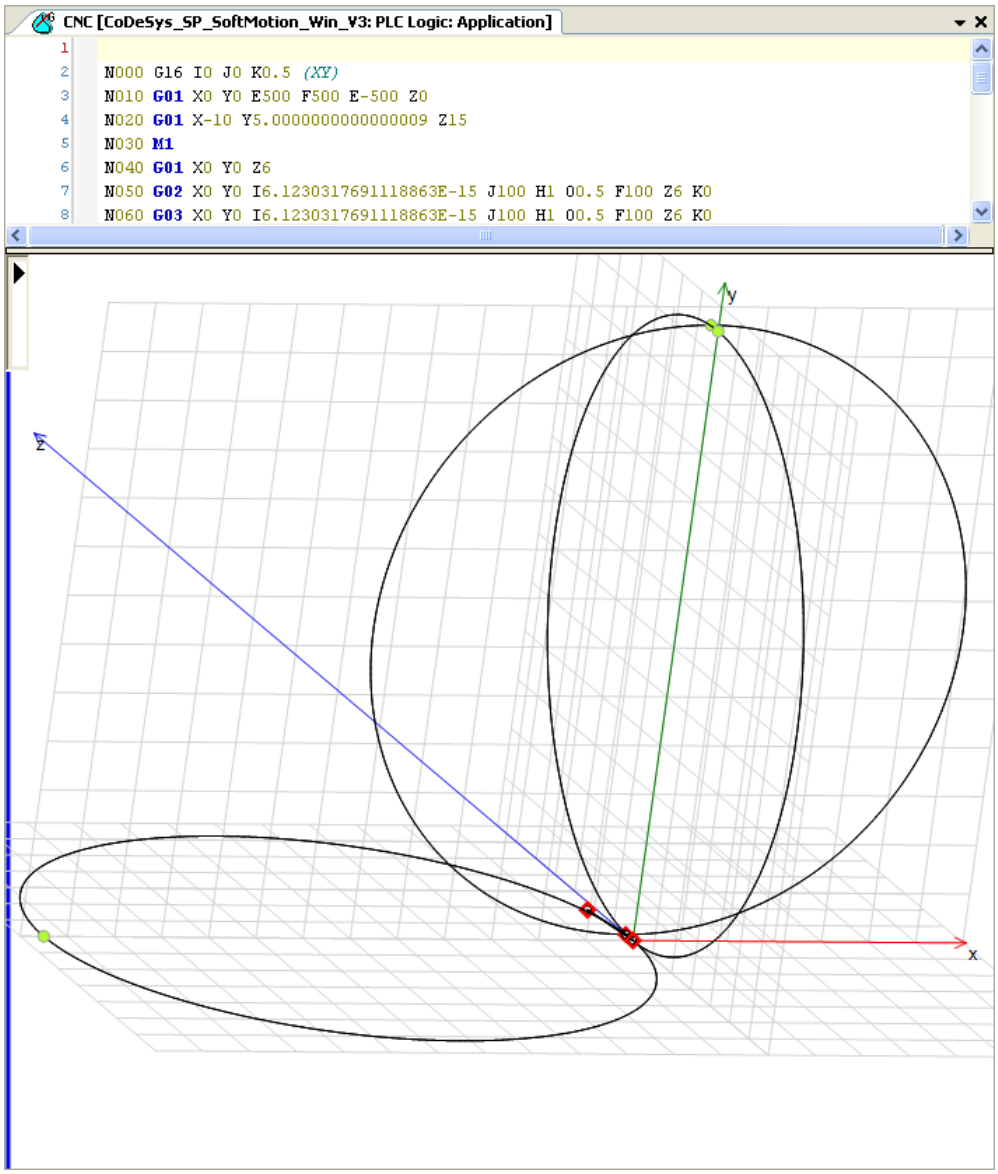


Abb. 327 CNC-Bahn im 3D-Modus

Ellipsen, Parabeln und 2D-Splines werden im 3D-Modus nicht unterstützt und eine Fehlermeldung wird generiert.

Der Funktionsblock SMC_CheckVelocities, der in der Bibliothek SM3_CNC zur Verfügung gestellt wird, überprüft die Z-Komponente im Unterschied zu den Funktionsblöcken SMC_ToolCorr und SMC_AvoidLoop, die Fehler generieren.

Kommando G15

Dieses Kommando wechselt für alle weiteren Elemente in den 2D-Modus.

Kommando G16

Dieses Kommando wechselt für alle weiteren Elemente in den 3D-Modus durch Aktivierung der 3D-Ebenenfunktion in der Normalenebene mit den Koordinaten I/J/K.

Kommando G17

Dieses Kommando wechselt für alle weiteren Elemente in den 3D-Modus durch Aktivierung der Ebenenfunktion in der X/Y-Ebene.

Kommando G18

Dieses Kommando wechselt für alle weiteren Elemente in den 3D-Modus durch Aktivierung der 3D-Ebenenfungktion in der Z/X-Ebene.

Kommando G19

Dieses Kommando wechselt für alle weiteren Elemente in den 3D-Modus durch Aktivierung der 3D-Ebenenfungktion in der Y/Z-Ebene.

Sprünge

Kommando G20

Mittels des Befehls G20 in DIN66025 kann ein bedingter Sprung realisiert werden. Dazu werden die Satznummer des Sprungziels (L) und die Bedingung für den Sprung (K) eingegeben. Wird keine Sprungbedingung angegeben, so wird automatisch die implizite Decoder-Variable verwendet. Der Sprung wird immer dann ausgeführt, wenn die Bedingung ungleich 0 ist.

Der grafische Editor unterstützt nicht die Verwendung der G36/G37 Befehle mit expliziten Variablen.

Erlaubte Syntax

G20 L O

Parameter des Befehls	
L	Zeilennummer des Sprungziels
O	Bedingung: Der Sprung wird ausgeführt, wenn der Wert von O nicht 0 ist (O<>0). Wenn O nicht angegeben ist, wird die interne Decoder-Variable verwendet.

Beispiel:

G-Code

```
N00 G36 D10       (Setzen des Zählers auf 1)
N10 G91          (relativer Modus)
N20 G01 X10 Y10 F100 (Bewegung um die Distanz 10/10)
N30 G37 D-1       (Dekrement-Zähler)
N40 G20 L20       (Sprung,wenn Zähler != 0)
```

Dieses Beispiel bewirkt, dass – durch den Relativmodus - zehn Geraden aneinandergehängt werden, d.h. es resultiert daraus eine Geradenbewegung auf 100/100.

Beispiel

Man beachte auch, dass die Sprung-Bedingung zum Zeitpunkt des Dekodierens ausgewertet wird. Sei im folgenden Beispiel die Variable x im Logik-Teil der Applikation so programmiert, dass sie FALSE wird, wenn die X-Achse die Position 15 überschreitet. Es leuchtet ein, dass nach dem ersten Durchlauf, da x initial auf TRUE steht und die X-Position erst 10 ist, wieder zurückgesprungen wird in Zeile 20. Beim zweiten Durchlauf könnte man nun meinen, dass die Bedingung für X=FALSE erfüllt sei, und kein Sprung zurück in Zeile 20 erfolgt, weil ja X 20 ist. Da die Auswertung der Bedingung aber zum Zeitpunkt des Dekodierens stattfindet, wenn der Interpolator noch nicht gestartet ist oder mit der Abarbeitung von Objekten vom Anfang des Buffer beschäftigt ist, wird die Schleife fortwährend durchlaufen. Erst wenn so viele Objekte erzeugt wurden, dass der Buffer voll ist und der Interpolator mit der Abarbeitung beginnt, würde diese Bedingung erfüllt werden und der Decoder aus dieser Schleife springen

G-code

```
N0 G92 X0 Y0
N10 G91          (Relativer Modus)
N20 G01 X10 Y10 F100 (Bewegung um die Distanz 10/10)
N30 G20 L20 O$x$     (Sprung, wenn Zähler != 0)
```

Abhilfe schaffen kann hier der Befehl Interpreterstop G75. Dieser bewirkt, dass der Decoder so lange mit der Verarbeitung pausiert, bis der Interpolator und mit ihm die Mechanik die entsprechende Stelle erreicht haben.

G-Code

```
N0 G92 X0 Y0
N10 G91          (Relativer Modus)
N20 G01 X10 Y10 F100 (Bewegung um die Distanz 10/10)
N25 G75
N30 G20 L20 O$x$     (Sprung, wenn Zähler != 0)
```

Ändern von Variablenwerten

Kommando G36, G37

Mit den Befehlen G36 und G37 steht dem Anwender die Möglichkeit zur Verfügung, Variablen zu schreiben oder zu verändern. Die Variable, die verändert werden soll, gibt der Anwender über Ovar an. Mit D gibt man den Wert vor, der auf die Variable geschrieben (G36) bzw. addiert (G37) werden soll.

Erlaubte Syntax

G36 O D

Parameter	
O	zu schreibende Variable. Wenn O nicht angegeben ist, wird die interne Decoder-Variable verwendet.
D	Neuer Variablenwert

G37 O D

Parameter	
O	zu inkrementierende Variable. Wenn O nicht angegeben ist, wird die interne Decoder-Variable verwendet.
D	Inkrement

Beispiel:

Folgendes Beispiel setzt die globale Variable g_i auf 5:

```
N1000 G36 O$g_i$ D5
```

Folgendes Beispiel bewirkt, dass die Zeilen 1010 und 1020 fünf Mal verfahren werden:

```
N1000 G36 O$g_i$ D5
N1010 G1 X100 F100 E100 E-100
N1020 G1 X0
N1030 G37 O$g_i$ D-1
N1040 G20 L1010 K$g_i$
```

Man beachte, dass dieser Mechanismus **nur funktioniert, wenn die Bahn online verarbeitet wird**, weil nur dann Variablen verwendet werden können! Im CNC-Editor funktioniert dieser Mechanismus also nicht.

Stattdessen kann folgendermaßen vorgegangen werden: Wird bei O keine Variable spezifiziert, so wird eine implizite Decoder-Variable (Typ: INT) verwendet. Dieser Mechanismus funktioniert auch offline im Editor. Man beachte aber, dass damit nur eine Variable zur Verfügung steht und man so keine verschachtelten Sprünge und Schleifen programmieren kann.

Vorverarbeitung

In DIN66025 müssen die vorverarbeitenden Module (SMC_SmoothPath, SMC_RoundPath, SMC_AvoidLoop, SMC_ToolCorr, anzuwählen in CNC-Einstellungen) mit G-Kommandos aktiviert zu werden. Dann werden die angefügten Bahnelemente bearbeitet und bei aktivierter Option Vorverarbeitete Bahn anzeigen im CNC-Menü ist die bearbeitete Bahn im grafischen Editor sichtbar.

Abb. 328 CNC Bahn mit Eckverschleifung

Befehle zur Werkzeugradiuskorrektur

Der Vorverarbeitungs-Funktionsblock kann mit folgenden Befehlen aufgerufen werden.

G40

Dieses Kommando beendet die Werkzeugradiuskorrektur.

G41

Dieses Kommando startet die Werkzeugradiuskorrektur links vom Werkstück.

G 42

Dieses Kommando startet die Werkzeugradiuskorrektur rechts vom Werkstück.

Zulässige Syntax

G40

G41 D

G42 D

Parameter des Befehls	
D	Werkzeugradius

Hinweis: Eine detailliertere Funktionsbeschreibung der Werkzeugradiuskorrektur befindet sich bei dem Vorverarbeitungsbaustein „SMC_TOOLCORR (FB)“ der Bibliothek „Bibliothek SM3_CNC.Bibliothek“

Befehle zur Eckverrundung und Eckverschleifung

G50

Dieses Kommando beendet die Eckverrundung (SMC_RoundPath) bzw. die Eckverschleifung (SMC_SmoothPath).

G51

Dieses Kommando startet die Eckverschleifung.

G52

Dieses Kommando startet die Eckverrundung.

Zulässige Syntax

G50

G51 D

G52 D

Parameter des Befehls	
D	Verschleifdistanz

Hinweis: Eine detailliertere Funktionsbeschreibung der Verschleifdistanz befindet sich bei dem Vorverarbeitungsbaustein „SMC_TOOLCORR (FB)“ der Bibliothek „Bibliothek SM3_CNC.Bibliothek“.

Beispiel

G-Code

```
N0 G51 D10 F100
N10 G01 X$g_x$ Y0 F50 E30 E-30
N20 G01 X0 Y$g_y$
N30 G01 X0 Y0
N40 G50
```

Befehle zur Schleifenvermeidung

G60

Dieses Kommando beendet die Funktion zur Schleifenvermeidung (SMC_AvoidLoop).

G61

Dieses Kommando startet die Funktion zur Schleifenvermeidung.

Zulässige Syntax

G60

G11

Hinweis: Eine detailliertere Funktionsbeschreibung der Schleifenvermeidung befindet sich bei dem Vorverarbeitungsbaustein „SMC_TOOLCORR (FB)“ der Bibliothek „Bibliothek SM3_CNC.Bibliothek“.

Koordinatenverschiebung

Die Funktionen G53/54/55/56 verändern den Positions-Offset, der die Position aller folgenden Kommandos verschiebt. Der Offset wird komplett im NCDecoder-Baustein gehandhabt. So werden die Koordinaten des Bahnobjektes (GeoInfo-Datenstruktur), die in der SMC_QutQueue des Decoders gespeichert sind, versetzt. Grundsätzlich arbeiten die Funktionen G54, 55 und 56 nicht nur mit den Achsen X/Y/Z, sondern auch mit allen zusätzlichen linearen Achsen ((P,Q,U,V,W).

Zugelassene Syntax

G53

G54 X Y Z A B C P Q U V W

G55 X Y Z A B C P Q U V W

G56 X Y Z A B C P Q U V W

Parameter des Befehls	
XYZA B C P Q U V W	Offset-Wert

Über die folgenden Kommandos der DIN66025 kann die Verschiebung des Koordinatensystems gehandhabt werden:

G53

Dieses Kommando beendet die Funktion zur Koordinatenverschiebung und das Koordinatensystems wird auf das ursprüngliche zurück gesetzt.

G54

Es setzt einen (absoluten) Offset im CNC-Programm (so, dass alle folgenden X/Y/Z-Koordinaten um die spezifizierte Position versetzt werden).

Beispiel:

```
N0 G0 X100 Y100
N0 G54 X50 Y50    (set offset 50/50)
N0 G1 X0 Y0       (movement to 50/50)
N0 G54 X100 Y100  (set offset 100/100)
N0 G1 X0 Y0       (movement to 100/100)
N0 G53            (set offset to 0)
N0 G1 X0 Y0       (movement to 0/0)
```

G55:

Zum bestehenden Offset wird X/Y/Z addiert.

Beispiel:

```
N0 G0 X100 Y100
N0 G54 X50 Y50   (set offset to 50/50)
N0 G1 X0 Y0      (movement to 50/50)
N0 G55 X100 Y100 (set offset to 150/150)
N0 G1 X0 Y0      (movement to 150/150)
N0 G53           (set offset to 0)
N0 G1 X0 Y0      (movement to 0/0)
```

G56 :

Der interne Offset wird im Decoder verändert, so dass die aktuelle Position X/Y/Z ist.

Beispiel:

```
N0 G0 X100 Y100
N10 G56 X0 Y0  (current position (100/100) becomes (0/0))
N20 G1 X10     (movement to (110/100))
N30 G56 X20 Y0 (current position (110/100) becomes (20/0))
N40 G1 X0      (movement to (90/100))
```

Hinweis: G54, G55, und G56 verändern den Offset nur für die Koordinaten, die als Parameter übergeben werden. Die übrigen Offset-Koordinaten bleiben unverändert.

Beispiel:

```
G54 X10 Y20 Z30  U7 (Offset: X=10, Y=20, Z=30, U=7)
G55 X-10 U7        (Offset: X=0, Y=20, Z=30, U=14)
```

Zeitsynchronisation mit Interpolator

Kommando G75

Dieses Kommando in DIN66025 verhindert, dass der Decoder-FB den G-Code abarbeitet, bis der Interpolator alle vorhergehenden Objekte abgefahren hat. Hat der Interpolator G75 erreicht, sind alle Queues (Warteschlangen) leer und der NCDecoder (und alle vorverarbeitenden Funktionsblöcke) laufen weiter.

Diese Funktion ist sinnvoll, wenn Variablen verwendet werden. Die Variablen werden vom Decoder ersetzt, wenn die korrespondierende Zeile abgearbeitet wird, aber die Ausführung vom Interpolator geschieht gemäß dem Warteschlangen-Mechanismus erst eine bestimmte Zeit später. G75 kann verwendet werden, um zu warten und die Evaluation der Variablen und die Weiterverarbeitung des Interpolators zu "synchronisieren".

Zulässige Syntax

G40

G75

Beispiel:

```
N10 G1 X100
N20 G75
N30 G1 Y$g_y$
```

In diesem Beispiel wird die Variable g_y im Element G10 zur Verfügung gestellt und ist vorher nicht bekannt. Wird G75 nicht benutzt, dann arbeitet der NCDecoder die Zeile 30 sofort ab, bevor der Wert bekannt ist .

G75 hat keine Auswirkung auf den CNC-Editor oder die Bahnen, die als SMC_OutQueue vom CNC-Editor generiert wird.

Modi

Kommando G90, G91, G98, G99

Diese Kommandos bestimmen, wie Koordinatenwerte interpretiert werden

G90

Dieses Kommando bewirkt die Auswertung der folgenden Koordinatenwerte (für X/Y/Z/A/B/C/P/Q/U/V/W) als absolute Werte. Es ist der standardmäßige Modus.

G91

Dieses Kommando startet die Auswertung der folgenden Koordinatenwerte (für X/Y/Z/A/B/C/P/Q/U/V/W) als relative Werte.

G98

Dieses Kommando bewirkt die Auswertung der folgenden Koordinatenwerte von I/J/K als absolute Werte.

G99

Dieses Kommando bewirkt die Auswertung der folgenden Koordinatenwerte von I/J/K als Werte relativ zum startenden Punkt. Es ist der standardmäßige Modus.

Zulässige Syntax

G90

G91

G98

G99

Beispiel

In diesem Beispiel bewegt sich der Interpolator zu 10/10 und weiter zu 100/10. Die Koordinaten werden als absolute Werte interpretiert.

G-Code

```
N0  G90
N10 G1 X10 Y10 F100 (Startposition)
N20 G1 X100 (Nächste Position)
```

Beispiel

In diesem Beispiel bewegt sich der Interpolator zu 10/10 und weiter zu 100/10. Die Koordinaten werden als relative Werte interpretiert

G-Code

```
N0  G91
N10 G1 X10 Y10 F100
N20 G1 X100
```

Beispiel

In diesem Beispiel befindet sich der Mittelpunkt des Halbkreises bei 150/0. Die Koordinaten werden als absolute Werte interpretiert.

G-Code

```
N00 G98
N10 G92 X100 Y0 (Startposition)
N20 G2 X200 Y0 I150 J0 F100
```

Beispiel

In diesem Beispiel befindet sich der Mittelpunkt des Halbkreises bei 150/0. Die Koordinaten werden als relative Werte interpretiert

G-Code

```
N00 G99
N10 G92 X100 Y0 (Startposition)
N20 G2 X200 Y0 I50 J0 F100
```

Schaltpunkt (H-Funktion)

Die Schaltpunkt-Funktionalität (oder H-Funktion) gibt dem CNC-Programmierer in DIN66025 die Möglichkeit, binäre wegabhängige Schalter zu bedienen.
Grundsätzlich muss immer zuerst die Nummer des Schaltpunkts spezifiziert werden (H<Nummer>), dann muss die Position des Schaltpunkts im Objekt entweder absolut über das Wort L<Position> oder relativ über das Wort O<Position> festgelegt werden.

Erlaubte Syntax

[G-Befehl] H O/L

Parameter des Kommandos	
H	Nummer der H-Funktion
O	Relative Position in Objekt [0..1] 0: Start 1: Ende
L	L> 0: Distanz zum Startpunkt L< 0: Distanz zum Endpunkt

H-Nummern

Ist die H-Nummer positiv, wird die entsprechende Schaltfunktion eingeschaltet, sonst wird sie abgeschaltet. H-Nummern dürfen nicht 0 sein und müssen im Bereich von -32767 bis 32767 liegen.

Die H-Nummern 1 bis 32 werden im Interpolator protokolliert. Der Ausgang dwSwitches des Bausteins SMC_Interpolator ist eine Bit-Maske. Bit Nummer i gibt an, ob die H-Funktion (i+1) momentan gesetzt ist oder nicht. Für H-Nummern, die größer als 32 sind wird nicht protokolliert.

Der Interpolator gibt im Ausgang iLastSwitch die zuletzt geschaltete H-Nummer aus. iLastSwitch ist positiv, wenn die H-Funktion eingeschaltet wurde und sonst negativ

Grafische Darstellung

Im grafischen Editor werden die Schaltpunkte mit einem kleinen, grün gefüllten Kreis auf der Bahn dargestellt. Sie können im grafischen Editor entlang der Bahn mit der Maus verschoben werden. Wird die Maus über die H-Funktion bewegt, wird ein Tool-Tipp mit der H-Nummer der H-Funktion angezeigt.

Beispiel:

Schaltpunkt 2 wird an der Position X=40/Y=25 (nach dem ersten Viertel des Objekts) abgschaltet.

```
N90 G1 X20 Y20
N100 G1 X100 Y40 H-2 O0.25
```

Beispiel

Schaltpunkt 2 wird an Position X=40 (20 Einheiten nach dem Startpunkt des Objekts) eingeschaltet und an Position X=90 (10 Einheiten vor dem Endpunkt)ausgeschaltet.

```
N90 G1 X20
N100 G1 X100 H2 L20 H-2 L-10
```

Hinweis: Für jedes Bahnobjekt kann nur eine begrenzte Anzahl an Schaltpunkten (MAX_SWITCHES) abgearbeitet werden. Standardwert: MAX_SWITCHES = 3.

M-Funktion

Mittels der Zusatz-Funktionen oder M-Funktionen in DIN 66025 kann ein binärer Ausganggesetzt werden, der eine anderweitige Aktion startet. Im Unterschied zu den Schaltpunkten wird dabei so lange an der aktuellen Position verharrt, bis die M-Funktion durch Setzen eines Eingangs am Interpolator bestätigt wird. Dies wird häufig verwendet, wenn die weitere Abarbeitung des Programms von anderen Prozessen abhängt.

Es ist möglich, zwei numerische Parameter, die das K- und L-Wort verwenden, zu übergeben. Werden mehr Parameter benötigt, dann kann eine Variable vom Typ SMC_M_PARAMETERS mit Ovar übergeben werden. Die Applikation kann die Werte dieser Parameter zur Laufzeit durch Verwendung der Funktion SMC_GetMParameters abfragen. Dabei werden alle verwendeten Parameter zum Zeitpunkt des Dekodierens ausgewertet und in der SMC_GEOINFO-Struktur im SMC_OUTQUEUE-Puffer gespeichert. Dies findet in der Regel eine gewisse Zeit vor der Ausführung der M-Funktion am Interpolator statt.

Zulässige Syntax

M K L O

Parameter des Befehls	
M	Nummer der M-Funktion. >0
K	Übergabe Parameter (LREAL)
L	Übergabe Parameter (LREAL)
O	Verweis auf Variable vom Typ SMC_M_PARAMETERS, die weitere Übergabe-Parameter enthält „SMC_M_PARAMETERS (STRUCT)“

Man beachte, dass die Bedeutung der einzelnen M-Funktionen vom System nicht festgelegt ist, sondern maschinenspezifisch durch die Applikation definiert und ausprogrammiert wird.

Grafische Darstellung:

Im grafischen Editor werden die Ausgänge durch eine kleine, dick rot eingefasste Raute auf der Bahn dargestellt. Die Ausgangsposition kann im CNC-Texteditor oder im Tabelleneditor eingegeben werden.

Beispiel

Folgende Zeile würde die M-Funktion 10 starten und so lange warten, bis diese bestätigt wird:

```
N90 M10 K100.7
```

Das folgende Beispiel setzt die selbst deklarierte Datenstruktur g_myMParams (Typ: SMC_M_PARAMETER) mittels Ovar. Mit einer Instanz des Bausteines SMC_GetMParameters kann man die Variablenwerte von K und der Struktur g_myParam zum Anstehzeitpunkt der M-Funkion wieder erhalten.

```
N150 M13 O$g_myMParam$
```

Eine M-Funktion bewirkt immer einen Bahnstop, d. h. der Interpolator bremst auf Geschwindigkeit 0, gibt am Ausgnag wM die Nummer der anstehenden M-Funktion aus und wartet auf deren Bestätigung in bAcknM, bevor wieder beschleunigt wird.

Mit dem Funktionsbaustein SMC_PreacknowledgeMFunction besteht die Möglichkeit, eine M-Funktion vorzeitig zu bestätigen, d. h. der Interpolator würde nicht anhalten, sondern – sofern die Funktion rechtzeitig ausgeführt wird – sich verhalten, als wäre die M-Anweisung nicht vorhanden.

Verwendung von Variablen

Im G-Code kann eine Variable oder ein IEC-Ausdruck eingefasst von zwei $-Zeichen verwendet werden, deren offline-Wert mit dem Befehl **Variable setzen** im CNC-Menü oder im Eigenschaften-Fenster des CNC-Objekts gesetzt werden kann.

Variablen oder Ausdrücke können nach fast allen Worten angegeben werden, einzige Ausnahme sind die Satznummern (N-Worte). IEC-Ausdrücke und Variablen werden erst im Online-Decoder-Baustein durch ihre momentanen Werte ersetzt. Dies geschieht, wenn die Zeile dekodiert wird, was einige Zeit vor der Abarbeitung des Elements im Decoder stattfindet.

Mit Ausnahme des O-Parameters bei M-Funktionen, der eine Variable vom Typ SMC_M_PARAMETER erwartet, müssen Variablen und Ausdrücke einen der folgenden IEC-Basistypen haben: INT, SINT, USINT, UINT, DINT, UDINT, BYTE, WORD, DWORD, REAL, LREAL.

Beispiel:

Ein Eintrag eines Arrays einer Struktur:

```
$CNC[1].X$.
```

In diesem Kontext ist zu beachten, dass das Verwenden von Variablen nur dann funktioniert, wenn das Programm als Programmvariable (SMC_CNC_REF) übersetzt und online vom Decoder-Baustein verarbeitet wird. Die Variablen werden zu dem Zeitpunkt ersetzt, an dem der Decoder die entsprechende Zeile verarbeitet.

Wird das CNC-Programm stattdessen als SMC_OutQueue übersetzt, funktioniert der Variablen-Mechanismus nicht, da die Bahn dann offline erzeugt und als unveränderliche Datenstruktur der Applikation übergeben wird. In diesem Fall – wie auch zur Offline-Anzeige – ersetzt der Editor Variablen mit deren Offline-Werten.

Die Verwendung von Variablen in Online gelesenen G-Code-Programmen benötigt zusätzliche Vorbereitung. Siehe SMC_VARLIST.

13.3.2 CNC-Tabelleneditor

Im Tabelleneditor werden die Bahnelemente in einer Tabelle aufgelistet. Die Spalten der Tabelle sind in Reihenfolge und Sichtbarkeit in CNC-Einstellungen unter dem Registerblatt **Tabelleneditor** konfigurierbar.

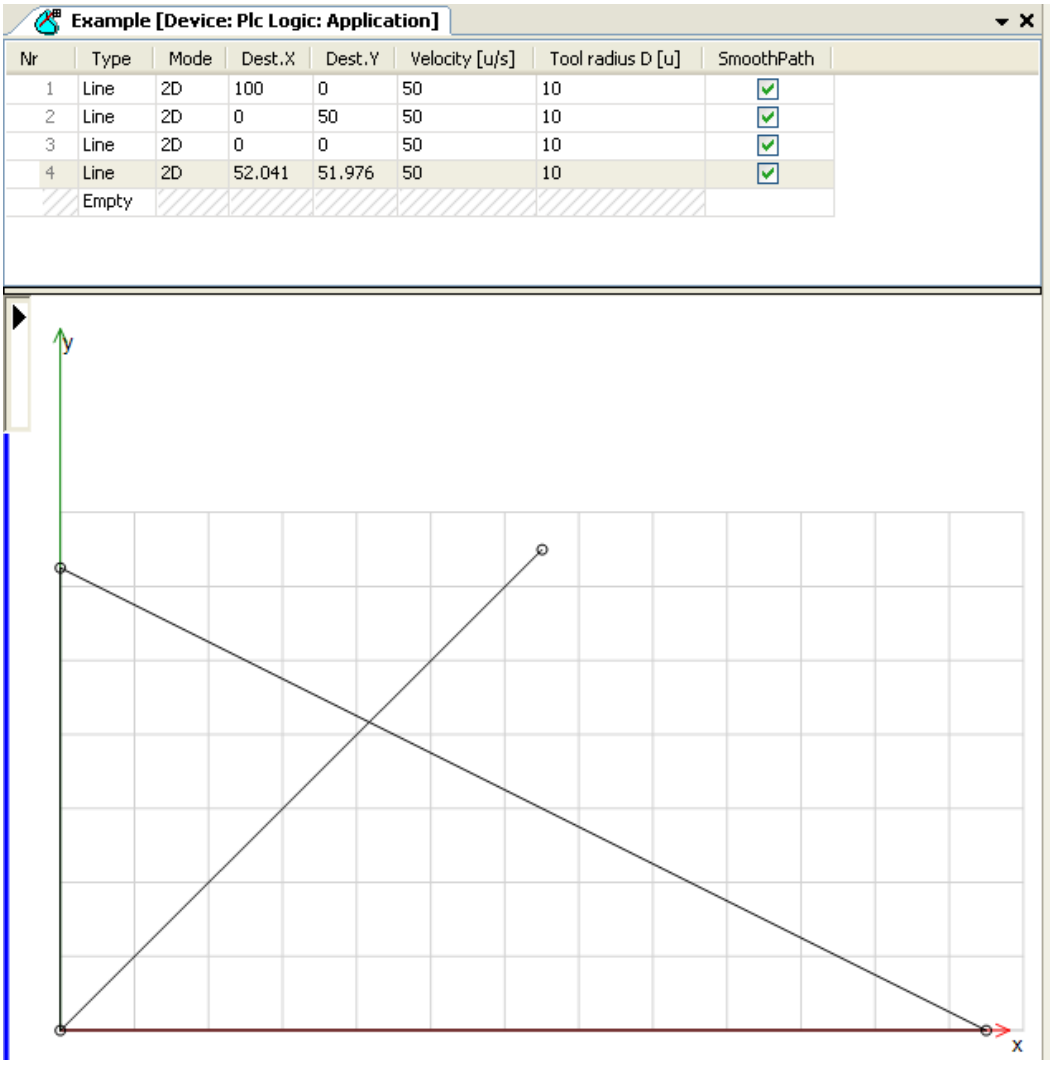


Abb. 329 CNC-Programm in Tabellendarstellung

Allgemein

Bei Klick in den Tabelleneditor wird die markierte Zeile **hellblau** hervorgehoben und der selektierte Tabelleneintrag wird durch einen dünnen, blauen Rahmen zusätzlich angezeigt. Das zugehörige Bahnelement wird dann rot mit einem blauen Startpunkt. Bei Klick in den grafischen Editor wird das aktuell selektierte Bahnelement in rot mit einem blauen Startpunkt dargestellt, während die entsprechende Zeile in der Tabelle **hellgrau** hervorgehoben wird. Die Bedienbarkeit der Zellen hängt von den Elementtypen ab. Eine Zelle, die für den Elementtyp nicht anwendbar ist, ist schraffiert und leer. Die Suchen-und-Ersetzen-Funktionalität wird nicht unterstützt.

Einfügen eines Elements

Die letzte Zeile der Tabelle ist eine leere und hat den Typ **Leer**. Wird auf diesen Eintrag doppelt geklickt und ein Typ des Pulldown-Menüs ausgewählt, dann wird eine neue Zeile angehängt. Ein weiterer Weg um ein neues Element zu bekommen ist, das Kontextmenü der Tabelleneinträge zu öffnen. Es stellt das Kommando **Element einfügen** zur Verfügung, das dann ein neues Element mit den gleichen Werten des zuvor eingefügten Elements in der Zeile vor der selektierten einfügt.

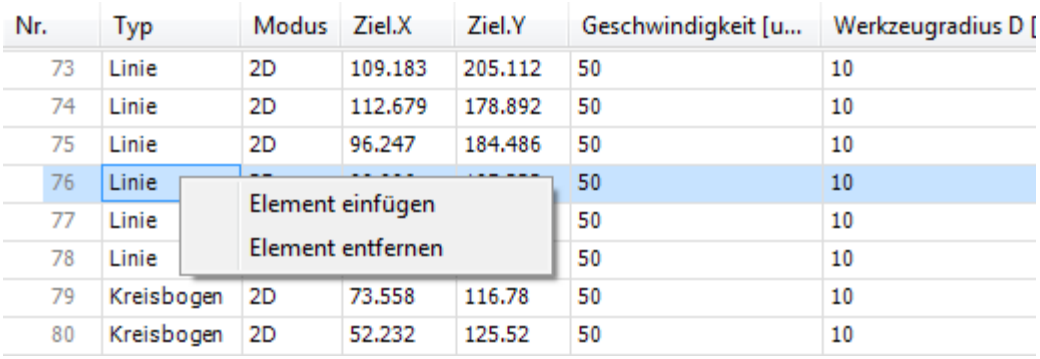


Abb. 330 Kontextmenü des Tabelleneditors

Ändern eines Tabelleneintrags

Der Inhalt der änderbaren Spalten wird in schwarzer Schrift angezeigt. Solche Einträge können durch einen Doppelklick auf sich selbst oder durch Selektieren mit der Maus und Drücken einer Taste editiert werden. Selektierte Einträge werden durch einen dünnen blauen Rahmen angezeigt. Dann öffnet der Zeileneditor oder ein Pulldown-Menü und der gewünschte Wert kann eingegeben werden. Zahlen werden im amerikanischen Stil formatiert. .(Die Nachkommastellen werden durch einen Punkt und nicht durch ein Komma vom ganzzahligen Teil getrennt.) Die Zeilennummer kann nicht editiert werden.

Im Kontextmenü der Tabelleneinträge wird das Kommando **Wert auf allen Elementen anwenden** zur Verfügung gestellt. Dieses Kommando kopiert die aktuellen Werte des Eintrags in alle anderen Einträge der selben Spalte, wenn der Wert für den jeweiligen Elementtyp anwendbar ist.

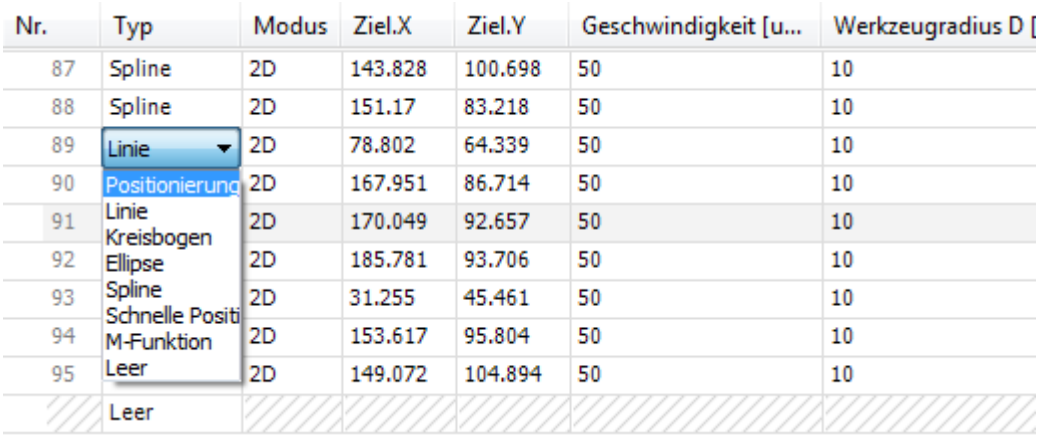


Abb. 331 Pulldown-Menü des Tabelleneditors

Löschen eines Elements

Eine selektierte Zeile kann mit der **Entf**-Taste oder durch Öffnen des Kontextmenüs und dem Kommando **Element entfernen** gelöscht werden.

Nicht unterstützte Elemente

Im Unterschied zum Texteditor unterstützt der Tabelleneditor die folgenden Elemente und Kommandos nicht: Parabel, IEC-Variablen, Sprünge, Koordinatenverschiebung, Satzunterdrückung, G4 und G75.

13.3.3 Grafischer Editor

Grafischer Editor (Pathview-Editor)

Der grafische Editor befindet sich im unteren Teil des CNC-Editors. Einerseits wird dort das CNC-Programm, das im Texteditor bzw. im Tabelleneditor beschrieben ist, dargestellt, andererseits kann das CNC-Programm auch mit Hilfe der ToolBox geändert werden. Alle Änderungen werden automatisch und unverzüglich im DIN66025-Editor bzw. in der Tabellenansicht im oberen Teil des Fensters aktualisiert.

Allgemein

Mit Klick in den grafischen Editor wird ein blauer Rand an der linken Seite des Fensters sichtbar, der den Window-Fokus anzeigt. Solange der Fokus auf dem grafischen Editor ist, wird der blaue Rand angezeigt. Mit Drücken von F6 wechselt der Fokus in den Text- oder Tabelleneditor und zurück. Ein Koordinatensystem mit grüner y-, roter x- und blauer z-Achse ist sichtbar. Ein hellgraues Gitter ist im Hintergrund, das über das Kommando **Raster anzeigen** im **CNC-**Menü an- und ausgeschaltet werden kann. Positionierungskommandos (G0) einschließlich Schaltpunkt-Funktionalität werden grün angezeigt. Im Unterschied dazu werden allen anderen Elementen schwarz angezeigt. Ist der Endpunkt verschiebbar, wird er als kleiner schwarz umrandeter Kreis dargestellt. Start- und Endtangente sind grau. Bei Klick auf ein Bahnelement wird es rot mit einem blau gefüllten Startpunkt und einem dünnen, schwarz gerahmten Endpunkt. Der entsprechende Satz im Code ist im Texteditor hellblau hinterlegt. Im Tabelleneditor ist die Zeile in hellgrau hervorgehoben. In der Statuszeile wird die aktuelle Position des selektierten Bahnelements angezeigt. Die Koordinaten der Mausposition werden, wenn möglich, in Weltkoordinaten angegeben.

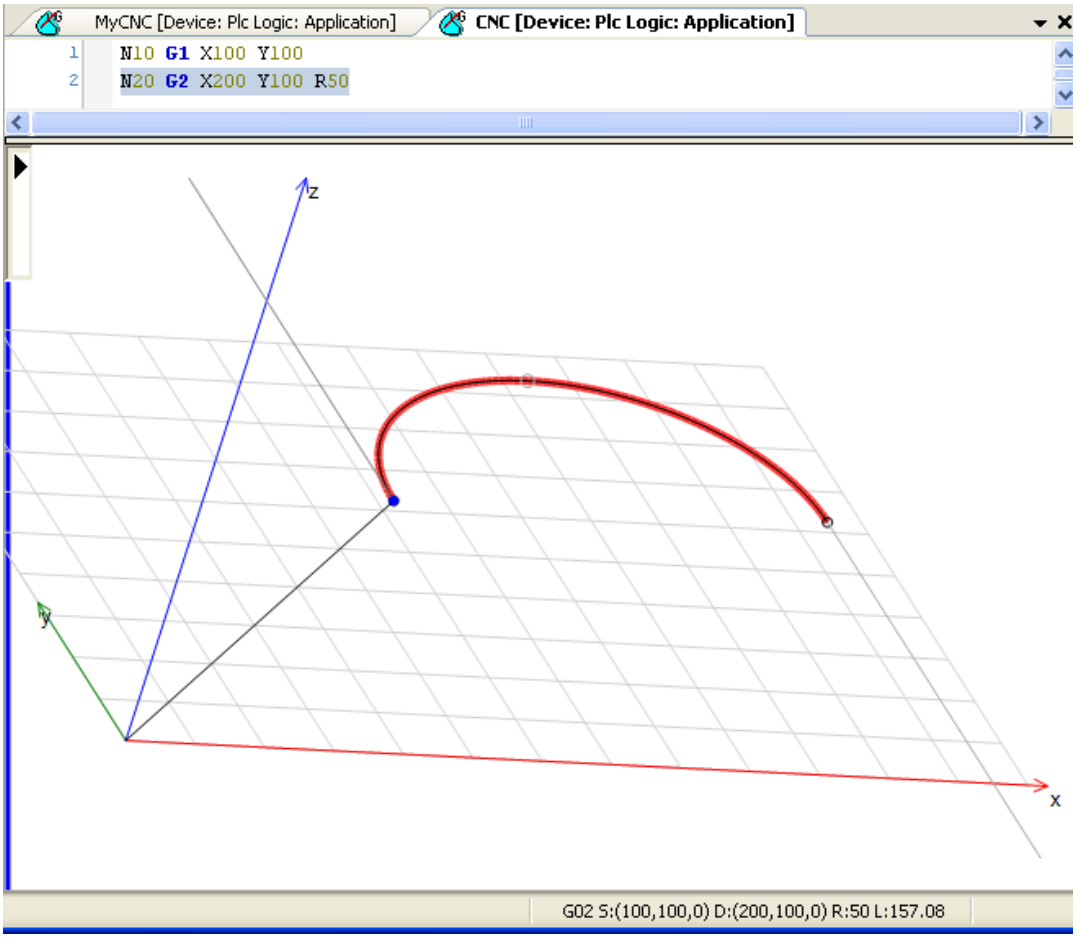


Abb. 332 Grafische Ansicht einer CNC-Bahn

Ändern der Kameraposition und Blickrichtung

Die Position und Blickrichtung der Kamera auf die Programmbahn kann mit Hilfe der Maus beliebig verändert werden:

- Mit gedrückter **linker Maustaste** wird die Kameraposition entsprechend der Mausbewegung in der Darstellungsebene entlang den Achsen geschoben.
- Mit gedrückter **rechter Maustaste** wird die Ansicht auf das CNC-Programm entsprechend der Mausbewegung gedreht.
- Drehen des **Mausrads** bei gleichzeitige gedrückter **Strg**-Taste führt zu einer Änderung des Zoomfaktors.

Das Bedienfeld im oberen linken Teil der grafischen Anzeige stellt ebenfalls verschiedene Kommandos zur Verfügung, um die CNC-Bahn zu bewegen.

Hinzufügen eines Bahnelements

Zum Einfügen oder Anhängen eines Bahnelements stehen im Fenster Werkzeuge verschiedene Kommandos zur Verfügung.

Ändern eines Bahnelements

Ein selektiertes Element kann durch Klicken auf die verschiebbaren Punkte, dargestellt durch kleine Kreise, geändert werden. Die Kreise werden dann dick rot. Sind Tangenten verschiebbar, werden sie nach einem Mausklick ebenfalls rot. Sie können dann geändert werden, indem ihre Start- und Endpunkte in der Darstellungsebene des selektierten Punktes verschoben werden. Abhängig vom Typ des Elements ist die Anzahl der verschiebbaren Punkte unterschiedlich. Der aktuelle Wert der zugehörigen Variablen ist gleichzeitig im CNC-Texteditor bzw. in der Tabellenansicht sichtbar.

Löschen eines Bahnelements

Ein selektiertes Element kann mit der **Entf**-Taste gelöscht werden.

Kommandos und Optionen

Verschiedenen Kommandos und Optionen stehen zur Verfügung, um zusätzliche Datensätze des CNC-Programms darzustellen oder diese zu manipulieren. Sie befinden sich im CNC-Menü.

Beispiele:

Verschiedene CNC-Programme werden mit dem Setup angefügt. Werfen sie einen genaueren Blick auf deren ausführliche Beschreibung.

Bedienfeld

In der linken oberen Ecke des grafischen Editors stellt ein Bedienfeld verschiedene Kommandos zur Verfügung um das Erscheinungsbild der Bahn anzupassen. Dabei wird die Bahn an sich nicht verändert, sondern die Kameraposition und Blickrichtung auf die Bahn wird verstellt.

Abb. 333 CNC Bedienfeld

Bei einem Klick in den schwarzen Pfeil verschwindet das Bedienfeld und die Ansicht auf die Bahn ist vergrößert. Ein kleiner Pfeil ermöglicht den Zugriff auf das Feld. Bei einem Klick auf das minimierte Feld vergrößert es sich und das gesamte Bedienfeld ist sichtbar.

Abb. 334 Minimiertes CNC Bedienfeld

Das Bedienfeld stellt folgende Befehle zur Verfügung um Orientierung, Position und Zoom einzustellen:

Vergrößern.

Verkleinern.

Rollwinkel der Kamera im Uhrzeigersinn verändern.

Rollwinkel der Kamera gegen den Uhrzeigersinn verändern.

Gierwinkel der Kamera im Uhrzeigersinn verändern.

Gierwinkel der Kamera gegen den Uhrzeigersinn verändern.

Kamera im Uhrzeigersinn rotieren.

Kamera gegen den Uhrzeigersinn rotieren.

Kamera nach oben bewegen.

Kamera nach links bewegen.

Kamera nach rechts bewegen.

Kamera nach unten bewegen.

Blickrichtung auf negative z-Achse setzen und neu skalieren.

Blickrichtung auf x-Achse setzen und neu skalieren.

Blickrichtung auf y-Achse setzen und neu skalieren.

13.3.4 Eigenschaften Bahnelemente

Die Eigenschaften eines aktuell im grafischen Editor des CNC-Editors selektierten CNC-Bahnelements werden in **Eigenschaften** dargestellt, das standardmäßig neben **Werkzeuge** zur Verfügung steht.

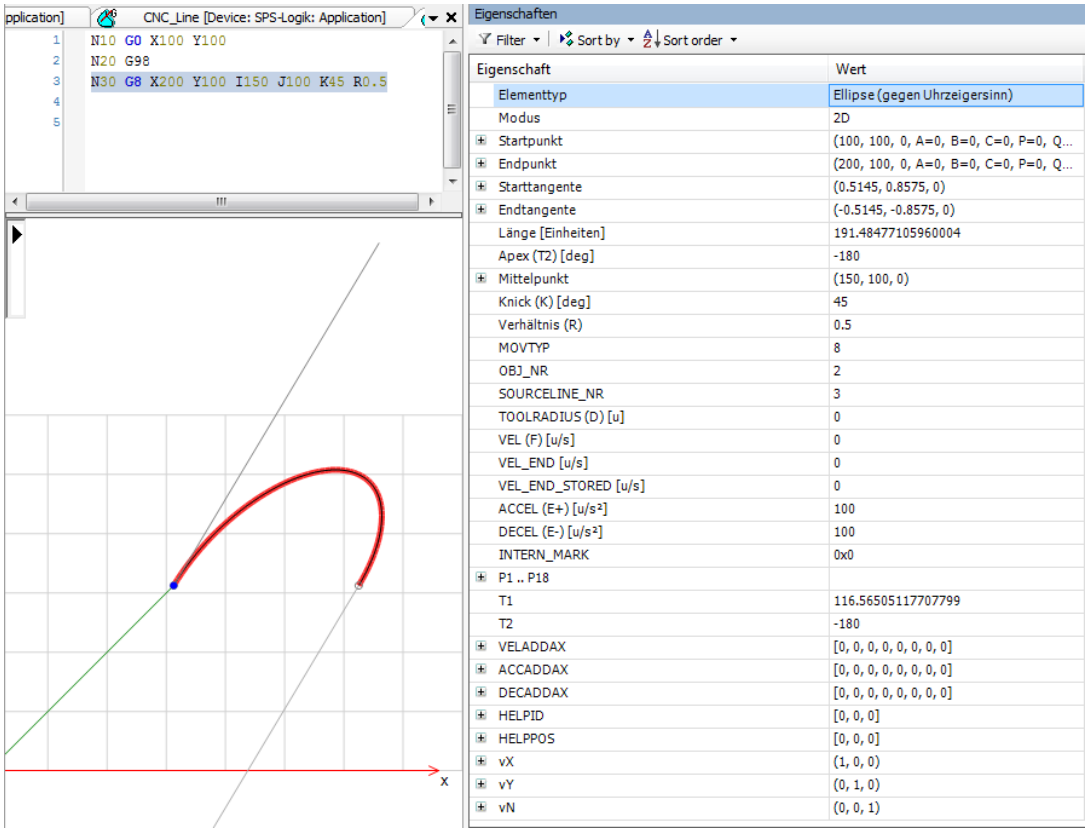


Abb. 335 Beispiel des Elementtyps Ellipse

13.3.5 Werkzeuge

Werkzeuge im CNC-Editor ermöglicht dem Anwender, Bewegungsbahnen im grafischen Editor zu programmieren und zu erweitern. Die Änderungen sind gleichzeitig im Texteditor und im Tabelleneditor sichtbar. Verschiedene Werkzeuge unterstützen das Programmieren der Bewegungsbahnen mit der Maus. Standard-Werkzeug ist **Auswählen**, das das Selektieren eines Bahnelements im grafischen Editor ermöglicht. Um in den Einfügen- bzw. Anhängen- Modus zu wechseln, ist auf eines der weiteren Werkzeuge zu klicken, je nachdem welches Element gewünscht wird.

- **Auswählen** ist standardmäßig aktiviert. In diesem Modus wird mit einem Mausklick im grafischen Editor ein Punkt oder ein Bahnelement selektiert, das dann rot mit einem blauen Startpunkt wird. Mit Klick auf wird ein inaktives Anwählen aktiv und der Cursor im grafischen Editor wird zu einem Zeiger.
- **Linien-Werkzeug** wird mit einem Klick auf aktiviert. Dann wird mit einem Klick in den grafischen Editor eine neue Linie eingefügt oder angehängt.

- Das **Kreis-Werkzeug (im Uhrzeigersinn)** wird mit einem Klick auf aktiviert. Dann wird mit einem Klick in den grafischen Editor ein neuer Kreisbogen im Uhrzeigersinn eingefügt oder angehängt.
- Das **Kreis-Werkzeug (gegen den Uhrzeigersinn)** wird mit einem Klick auf aktiviert. Dann wird mit einem Klick in den grafischen Editor ein neuer Kreisbogen gegen den Uhrzeigersinn eingefügt oder angehängt.
- Das **Spline-Werkzeug** wird mit einem Klick auf aktiviert, woraufhin mit einem Mausklick in den grafischen Editor ein neuer Spline eingefügt oder angehängt wird.

Einfügen eines Bahnelements

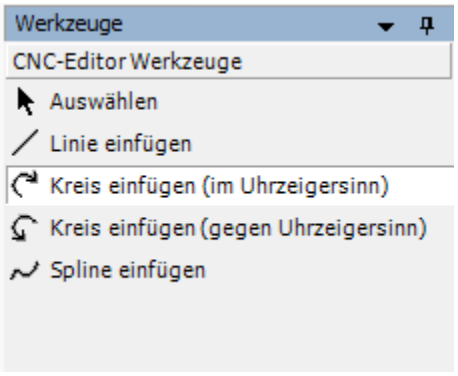


Abb. 336 CNC Editor Werkzeuge im Einfügen-Modus

Um ein Bahnelement einzufügen, ist im grafischen Editor ein Element zu selektieren. **Werkzeuge** befindet sich nun im Einfüge-Modus. Dann wird der gewünschte Elementtyp im Werkzeuge-Editor selektiert. Mit einem Klick in den grafischen Editor wird der Endpunkt des neuen Elements festgelegt. Startpunkt ist der Startpunkt des bisherigen selektierten Elements. Nach dem Einfügen ist das zuvor selektierte Element immer noch selektiert. Der Werkzeuge-Editor bleibt dabei im Einfügemodus, so dass durch weitere Mausklicks auf einfache Art mehrere Elemente desselben Typs vor dem selektierten Element eingefügt werden können.

Anhängen eines Bahnelements

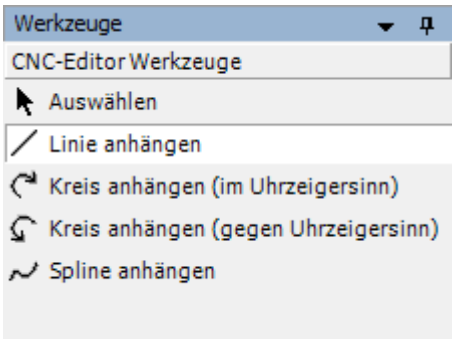


Abb. 337 CNC Editor Werkzeuge im Anhängen-Modus

Um ein Bahnelement anzuhängen darf kein Element im Pathview-Editor selektiert sein. Mit Klick auf den gewünschten Elementtypen in Werkzeuge befindet sich dieses im Anhänge-Modus. Jeder weitere Klick im Pathview-Edior markiert einen neuen Endpunkt der Bahn und ein neues Element wird angehängt. Es wird in schwarz mit zwei kleinen schwarzen Kreisen für Start- und Endpunkt dargestellt.

Allgemein

Im Einfügen- oder Anhängen-Modus ist der Cursor im grafischen Editor durch ein Kreuz repräsentiert. Der Modus wechselt in den Select-Modus, wenn die ESC-Taste verwendet wird. Dann wird der Cursor wieder ein Zeiger.

Falls der gewählte Übersetzungsmodus Texteditor ist, wird in diesem Editor simultan der Code mit folgender Farbgebung erweitert: Der zugehörige Satz im Texteditor ist in hellblau hervorgehoben.

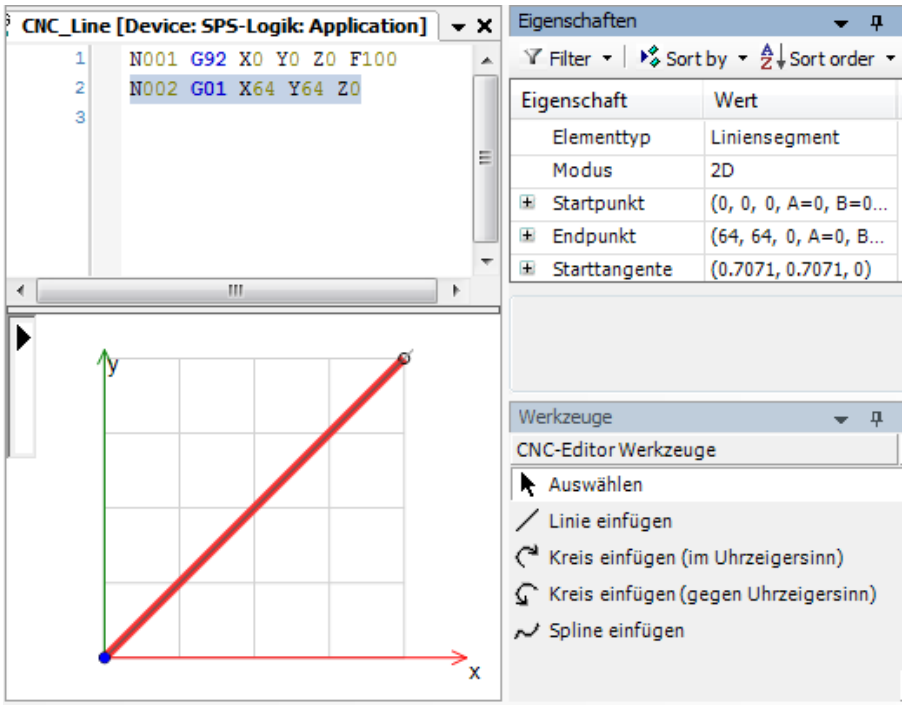


Abb. 338 CNC-Programm in DIN66250

Falls der gewählte Übersetzungsmodus der Tabelleneditor ist, wird in der Tabelle simultan der Code mit folgender Farbgebung erweitert: Die Zeile im Tabelleneditor ist in hellgrau hervorgehoben.

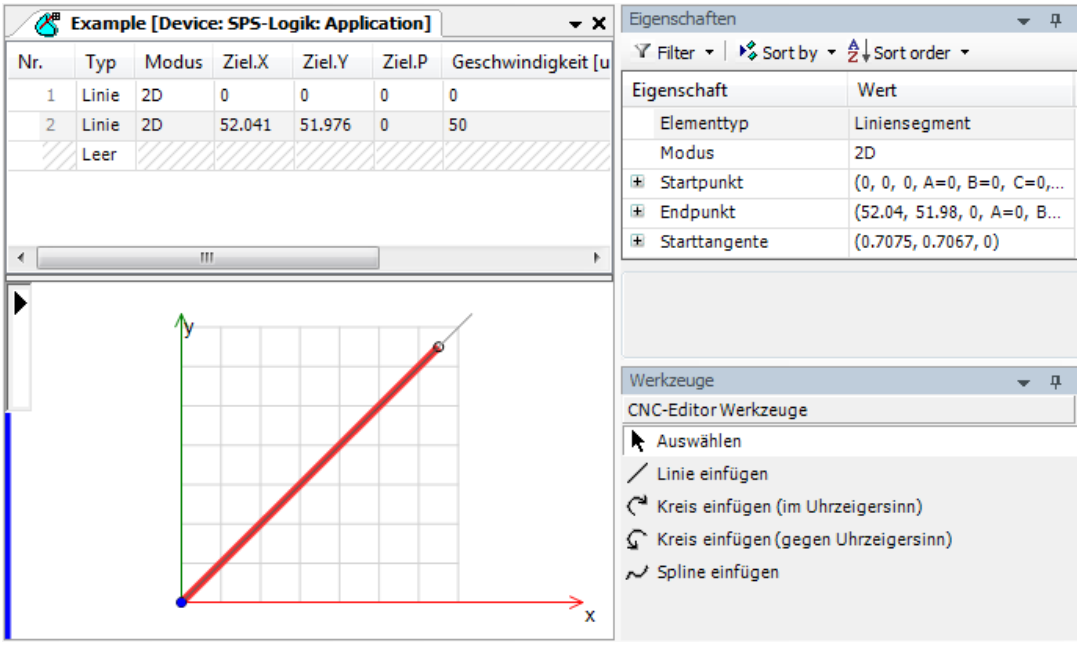


Abb. 339 CNC-Programm in Tabellenansicht

13.3.6 CNC-Objekt

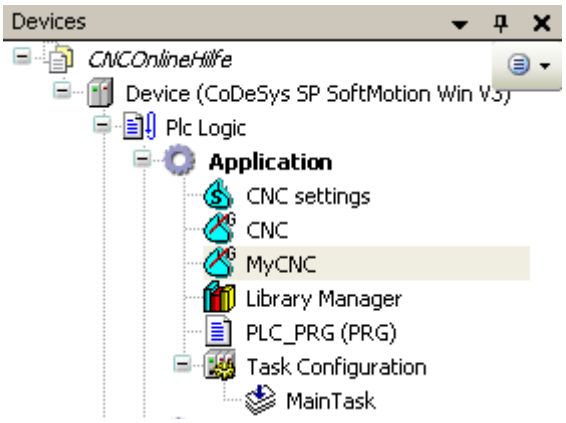


Abb. 340 Gerätebaum mit CNC-Objekt

Neues Objekt:

CNC-Objekte können nur unterhalb von SoftMotion- fähigen Steuerungen angelegt werden. Mit Auswahl der Applikation und der Aktivierung des Eintrags **Objekt hinzufügen** des Kontextmenüs wird der zugehörige Dialog geöffnet. Dort sind ein treffender Name, die gewünschte Art der Implementierung (DIn66025 oder Tabelle) und der Übersetzungsmodus einzugeben. Mit Klicken der OK-Taste wird dann ein neues Objekt im Gerätebaum eingefügt. Außerdem wird das Objekt **CNC Einstellungen** generiert, falls ein solches noch nicht vorhanden ist. Dort sind die applikationsweiten CNC-Einstellungen gespeichert. Die objektzugehörigen Einstellungen werden mit dem CNC-Objekt gespeichert und können im Dialog **Eigenschaften...** des CNC-Objekts editiert werden.

Objekt löschen:

So wie bei allen anderen Objekten ist es möglich, das CNC-Objekt zu löschen.

Kontextmenü:

Mit Öffnen des Kontextmenüs des CNC-Objekts stehen zusätzlich die Dialoge Eigenschaften... des CNC-Objekts und CNC Info zur Verfügung.

Eigenschaften des CNC-Objekts

Mit dem Befehl **Eigenschaften...** des CNC-Objekts öffnet den **Eigenschaften-CNC**-Dialog, wo spezifische Eigenschaften des CNC-Programms geändert werden können. Der **Eigenschaften...**-Befehl ist im Kontextmenü verfügbar oder in der **Ansicht**-Kategorie zu finden. Der Dialog stellt vier Registerkarten zur Verfügung: Allgemein, CNC, Zugriffskontrolle und Build.

Allgemein

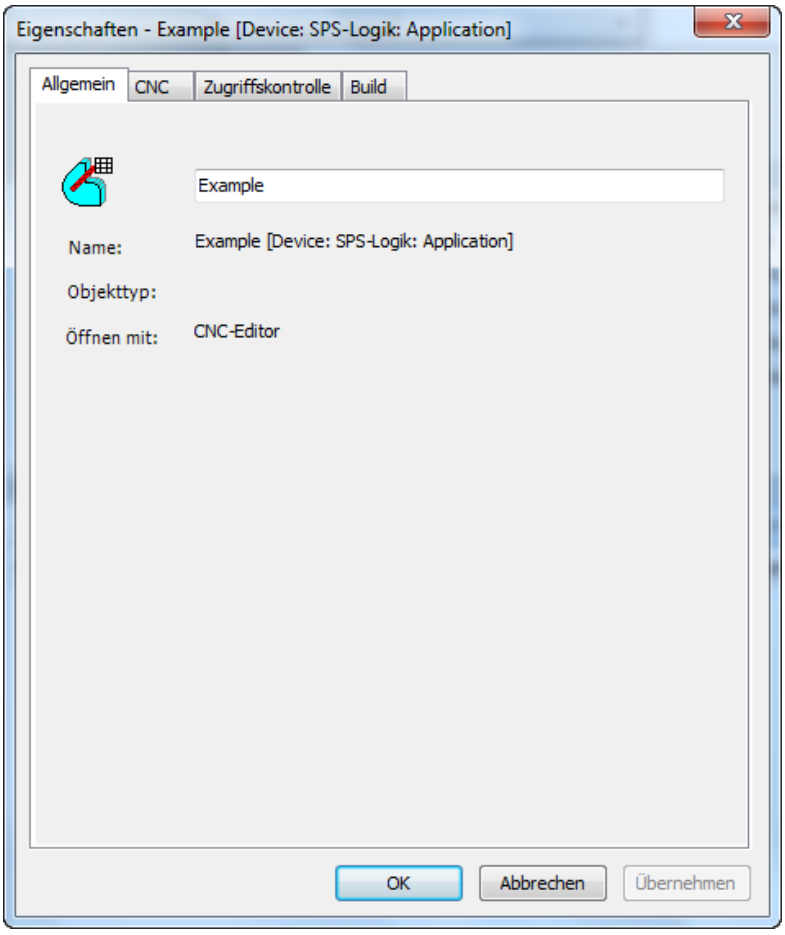


Abb. 341 Eigenschaften, Allgemein

Der Name des CNC-Objekts kann hier geändert werden.

CNC

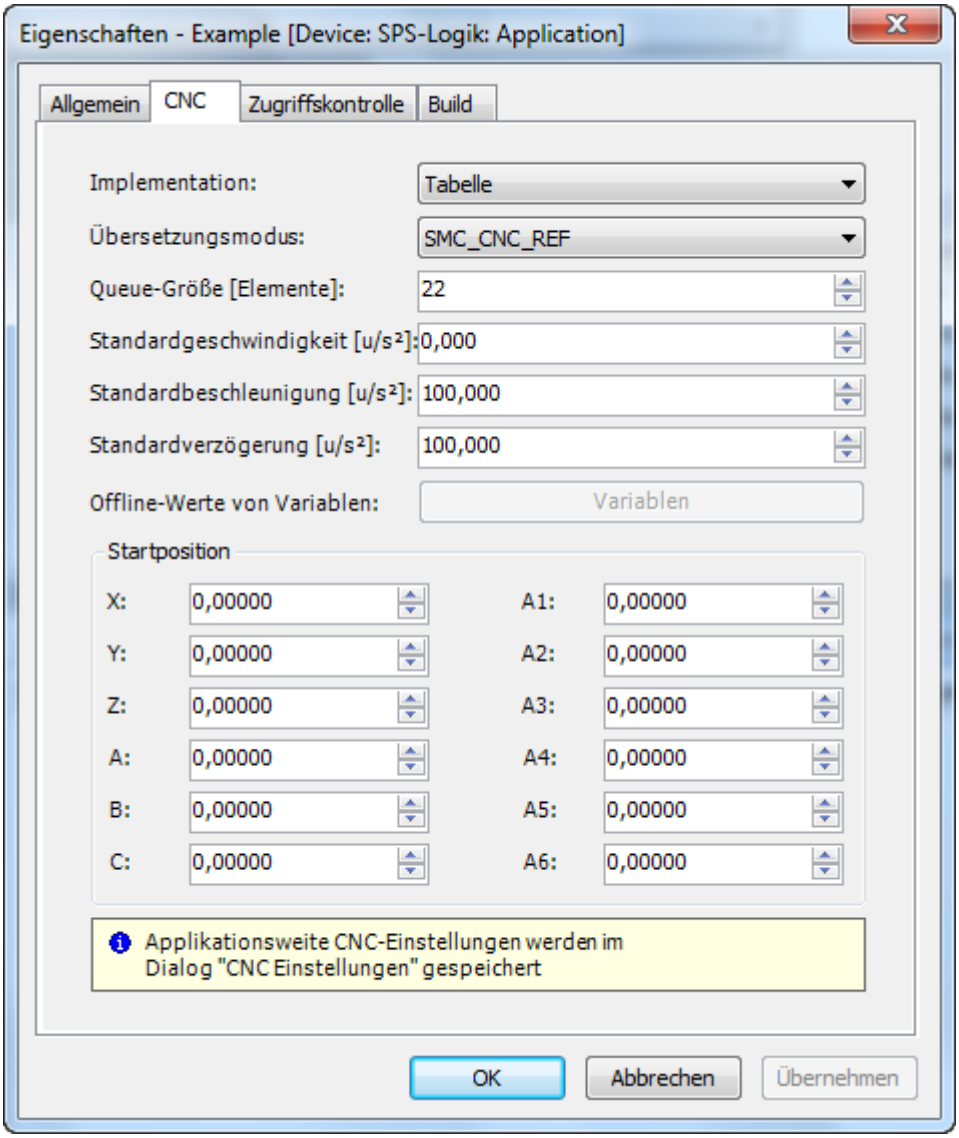


Abb. 342 Eigenschaften, CNC

- **Implementation:**
  Hier wird der Implementationstyp festgelegt.
  - Din66025
  - Tabelle

- **Übersetzungsmodus:**

  Siehe Kompilieren des CNC Programms.

  - SMC_OutQueue: Erstellen der SMC_OutQueue-Daten beim Kompilieren. Globaler Zugriff auf diese Daten ist im IEC-Programm über **g_CNCQueueManager** möglich.
  - SMC_CNC_REF: Erstellen von Programm-Variablen beim Kompilieren. Globaler Zugriff auf diese Daten ist im IEC-Programm über **g_CNCProgManager** möglich.
  - **Datei**: Durch Auswahl dieser Option wird eine Datei mit dem programmierten C-Code erzeugt und durch ein Einloggen in die SPS geladen. Dort kann die Applikation die Bahn aus der Datei lesen und ausführen. Dies erfolgt mit Hilfe von Funktionsblöcken wie SMC_NCDecoder und anderen aus der Bibliothek SM3_CNC. Die G-Code Dateien in der Steuerung werden bei jedem Login durch die G-code Objekte der Applikation aktualisiert. Dies bedeutet, dass jede Änderung der Dateien auf der Steuerung mit einem neuen Downlaod überschrieben werden.

- File name: Wenn als Übersetzungsmodus Datei ausgewählt ist, dann kann hier der Dateiname, welcher verwendet werden soll, eingetragen werden. Der Standardname ist $ObjectName$.cnc, welcher als Platzhalter dient und durch den Objektnamen des CNC-Objektes (tatsächlicher Dateiname) beim Einloggen ersetzt wird. Falls Sie den Dateinamen ändern, beachten Sie bitte die zulässigen Zeichen [a-zA-Z0-9$_.] für Dateinamen. Das Feld für den Dateinamen darf werden leer sein oder mit einem Punkt anfangen oder enden. Weiterhin muss ein Punkt enthalten sein und im Dateinamen darf kein Pfad enthalten sein.
- **Queue-Größe:** Standardwert ist 100.
- **Standardgeschwindigkeit:** Standardwert ist 0. Dieser Wert wird verwendet, falls keine Geschwindigkeit eines Bahnelements im CNC-Programm gesetzt ist. Wertebereich: [0, 1e9]
- **Standardbeschleunigung:** Standardwert ist 100,000. Dieser Wert wird verwendet, falls keine Beschleunigung eines Bahnelements im CNC-Programm gesetzt ist. Wertebereich:]0, 1e9].
- **Standardverzögerung:** Standardwert ist 1000,000. Dieser Wert wird verwendet, falls keine Verzögerung eines Bahnelements im CNC-Programm gesetzt ist. Wertebereich:]0, 1e9].
- **Startposition:** Standardwert ist 0.
- **Offline-Werte von Variablen:** Falls Offline-Variablen definiert wurden, können sie mit Klick auf die **Variablen**-Schaltfläche angezeigt und editiert werden. Das ist auch über den Befehl **Variablen setzen** möglich.

Die applikationsweiten CNC-Einstellungen werden im Objekt CNC-Einstellungen gespeichert.

Zugriffskontrolle

Hier können die Rechte der Benutzergruppen geändert werden.

Build

Hier kann das CNC-Objekt vom Buildprozess ausgeschlossen werden. Es werden dann keine IEC-Daten generiert und es ist möglich, das Programm als ASCII- oder als SMC_OutQueue-Datei im Dateisystem der Steuerung zu speichern. Falls gewünscht wird, das CNC-Programm zur Laufzeit einzulesen, ist ein Baustein der Bibliothek SM3_CNC zu verwenden.

CNC-Info

Der Dialog **CNC-Information von Programm <CNC Programmname>** stellt folgende Information über das Programm, das aktuell im Editor geladen ist, zur Verfügung: Programmname, Version der Funktionsbausteine, Anzahl der Zeilen, Anzahl der Objekte, Bahnlänge und Dauer. Bei erfolgreich generierter Bahnvorverarbeitung und erfolgreich generierten Interpolationspunkten wird die Dauer für die Bahn inklusive Vorverarbeitung angezeigt.

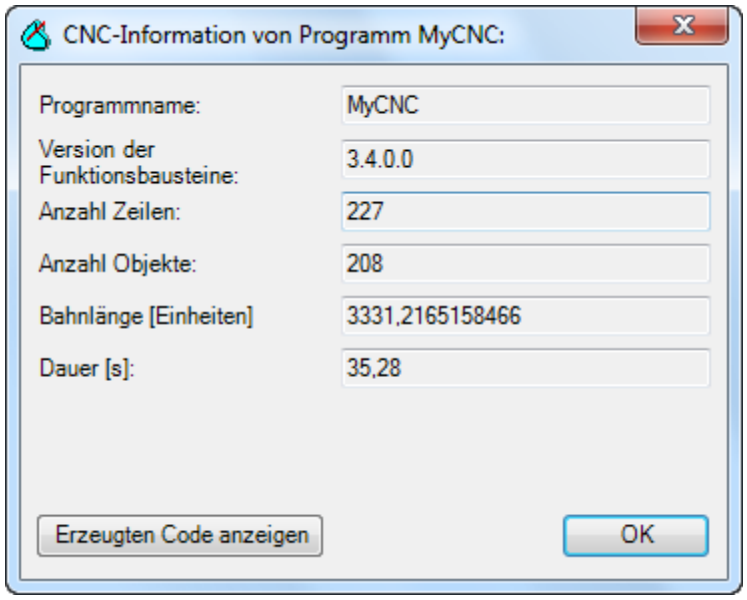


Abb. 343 CNC-Info des Programms MyCNC

Erzeugten Code anzeigen

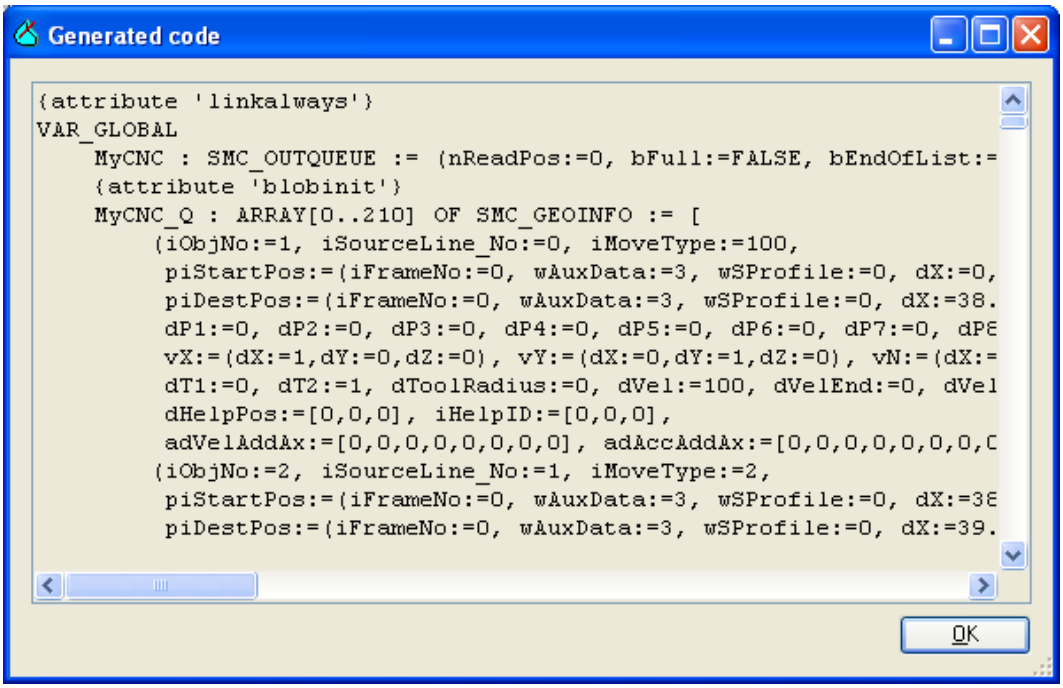


Abb. 344 Dialog "Erzeugter Code"

Bei Verwendung der Schaltfläche **Erzeugten Code anzeigen** wird ein Fenster mit der Auflistung der programmierten Bahn des zugehörigen CNC-Objekts angezeigt. Das kann nur angezeigt werden, wenn das CNC-Programm fehlerfrei ist. Dann kann man den generierten ST-Code des CNC-Programms sehen, und zwar den Deklarationsteil, der die SMC_OutQueue beziehungsweise die SMC_CNC_REF initialisiert.

13.3.7 CNC-Datenstrukturen

Im Zuge der Kompilierung des Projekts werden die erstellten CNC-Daten intern in eine globale Variablenliste umgewandelt ("CNC-Daten"). Ist der Übersetzungsmodus SMC_OutQueue wird jedes CNC-Element durch eine Strukturvariable vom Typ **SMC_OUTQUEUE** repräsentiert, ist er SMC_CNC_REF wird ein CNC-Element durch **SMC_CNC_REF** repräsentiert. Die Datenstrukturen werden während dem Ablauf des IEC-Programms instanziert und mit Werten gefüllt, die dann in IEC-Programmen verarbeitet werden können. Dazu wird eine globale Variable zur Verfügung gestellt. Alle Strukturen sind Teil der SM3_CNC Bibliothek.

Globaler Zugriff auf CNC-Daten

Während der Initialisierung werden Typ und Anzahl der generierten Objekte aus dem CNC-Programm ermittelt. In einer implizit deklarierten globalen Datenstruktur werden all diese Objekte zusammen aufgelistet. Objekte vom Typ **SMC_OUTQUEUE** werden in dem globalen Datenobjekt **g_CNCQueueManager** mit der Eigenschaft **Count** und der Methode **GetOutQueue(int n)** aufgelistet. Objekte vom Typ **SMC_CNC_REF** werden in dem globalen Datenobjekt **g_CNCProgManager** mit der Eigenschaft **Count** und der Methode **GetProgram(int n)** aufgelistet. Der Anwender kann dann in einem IEC-Programm darauf zugreifen.

Beispiel:

Zugriff auf Datenobjekte der Struktur **SMC_OUTQUEUE:**

```
PROGRAM CNCManageQueue
VAR
   pCNC_OutQueue  : POINTER TO SMC_OUTQUEUE;
   n: INT;
   i: INT;
END_VAR

n := g_CNCQueueManager.Count;
FOR i:=0 TO n-1 DO
pCNC_OutQueue := g_CNCQueueManager.GetOutQueue(i);
 (* pCNC_OutQueue verarbeiten *)
END_FOR
```

Zugriff auf Datenobjekte der Struktur **SMC_CNC_REF:**

```
PROGRAM CNCManageRef
VAR
   pCNC_Ref: POINTER TO SMC_CNC_REF;
   n: INT;
   i: INT;
END_VAR

n := g_CNCProgManager.Count;
FOR i:=0 TO n-1 DO
pCNC_Ref  := g_CNCProgManager.GetProgram(i);
 (* pCNC_Ref verarbeiten *)
END_FOR
```

13.3.8 CNC Online

Im Online-Modus ist keine Statuszeile sichtbar.

13.4 CNC-Einstellungen-Editor

Symbol:

Das Objekt **CNC-Einstellungen** wird automatisch erzeugt, wenn ein CNC-Objekt generiert wird und kann nicht umbenannt werden. Es befindet sich im Gerätebaum unterhalb einer Applikation. Es darf pro Applikation nur einmal existieren. Die dort gespeicherten Einstellungen sind für alle CNC-Objekte der Applikation gültig. Einstellungen, die nur für ein bestimmtes CNC-Programm gültig sind, werden im Eigenschaften-Dialog des CNC-Programms gespeichert.

Ein Doppelklick auf das CNC-Einstellungen-Objekt öffnet einen Dialog. Dort können Einstellungen für die bahnvorverarbeitenden Module, die Vorinterpolation und den CNC-Tabelleneditor festgelegt werden. Das Objekt kann nur gelöscht werden, wenn kein CNC-Objekt mehr unter der Applikation existiert.

Bahnvorverarbeitende Module

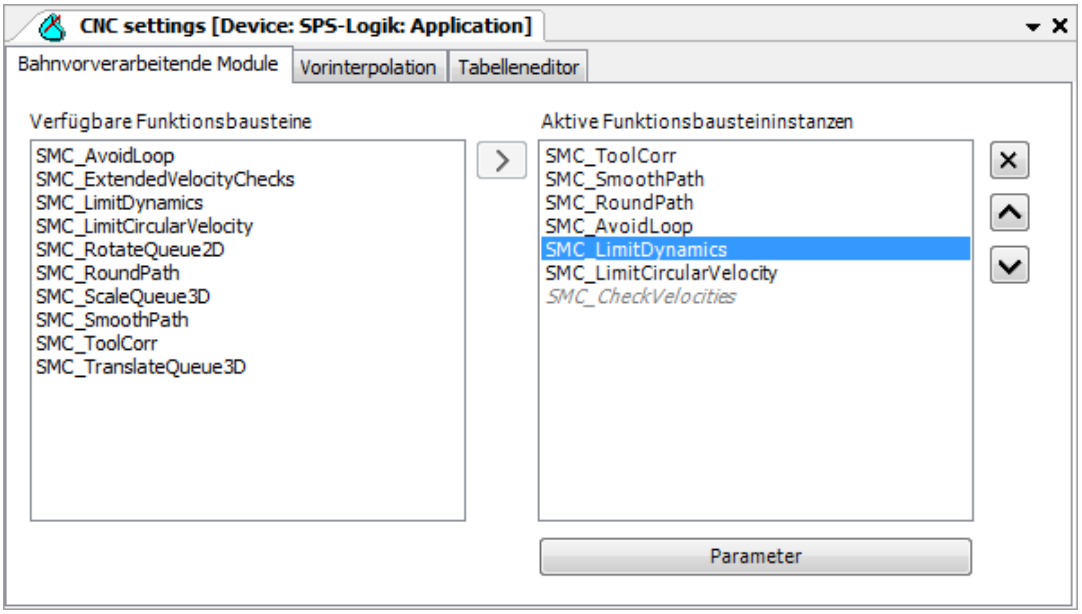


Abb. 345 CNC-Einstellungen für Bahnvorverarbeitung

Hier kann die Bahnvorverarbeitung durch Auswahl der gewünschten Funktionsblöcke programmiert werden. Die Position in der Liste der aktiven Funktionsblöcken definiert die Reihenfolge der Abarbeitung und kann geändert werden. Es ist möglich, den gleichen Funktionsblock mehr als einmal in der Liste der aktiven Funktionsblock-Instanzen zu instanziieren.

Im linken Teil werden die verfügbaren Funktionsblöcke aufgelistet, im rechten Teil die aktiven Instanzen der Funktionsblöcke. Es ist nicht möglich, Instanzen von Funktionsblöcken zu löschen, die in kursiver, hellgrauer Schrift angezeigt werden.

Diese Schaltfläche oder die **Leertaste** wird verwendet, um einen selektierten und verfügbaren Funktionsblock zu instanziieren, Das Gleiche kann mit einem Doppelklick auf den gewünschten Funktionsblock erreicht werden.

Diese Schaltfläche oder die **Entf**-Taste wird verwendet, um die selektierte Instanz zu löschen.

Diese Schaltfläche wird verwendet, um eine selektierte Instanz um eine Position nach oben zu schieben. Das Gleiche kann erreicht werden, wenn die **Nach-Oben-Cursortaste** mit gleichzeitig gedrückter **Umschalt-Taste** benutzt wird.

Diese Schaltfläche wird verwendet, um eine selektierte Instanz um eine Position nach unten zu schieben. Das Gleiche kann erreicht werden, wenn die **Nach-Unten-Cursortaste** mit gleichzeitig gedrückter **Umschalt-Taste** benutzt wird.

Einige Instanzen haben Parameter, die bei Klicken auf den Button **Parameter** oder bei einem Doppelklick auf die Instanz gesetzt oder geändert werden können, um den Dialog **Parameter-Editor** zu öffnen. Das Gleiche kann mit **Alt + P** erreicht werden.

Vorinterpolation

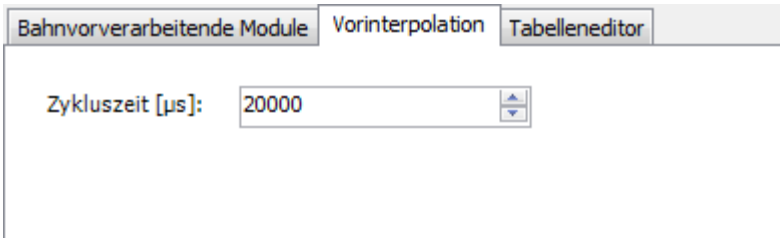


Abb. 346 CNC-Einstellungen für Vorinterpolation

Hier kann die Interpolationszeit gesetzt werden. Standard-Wert ist 20000 µs.

Hinweis: Interpolationszeit und Zykluszeit der zugehörigen Task, in der der Interpolator aufgerufen wird, müssen für ein identisches Verhalten von CNC-Editor und Online-Applikation zusammenpassen.

Tabelleneditor

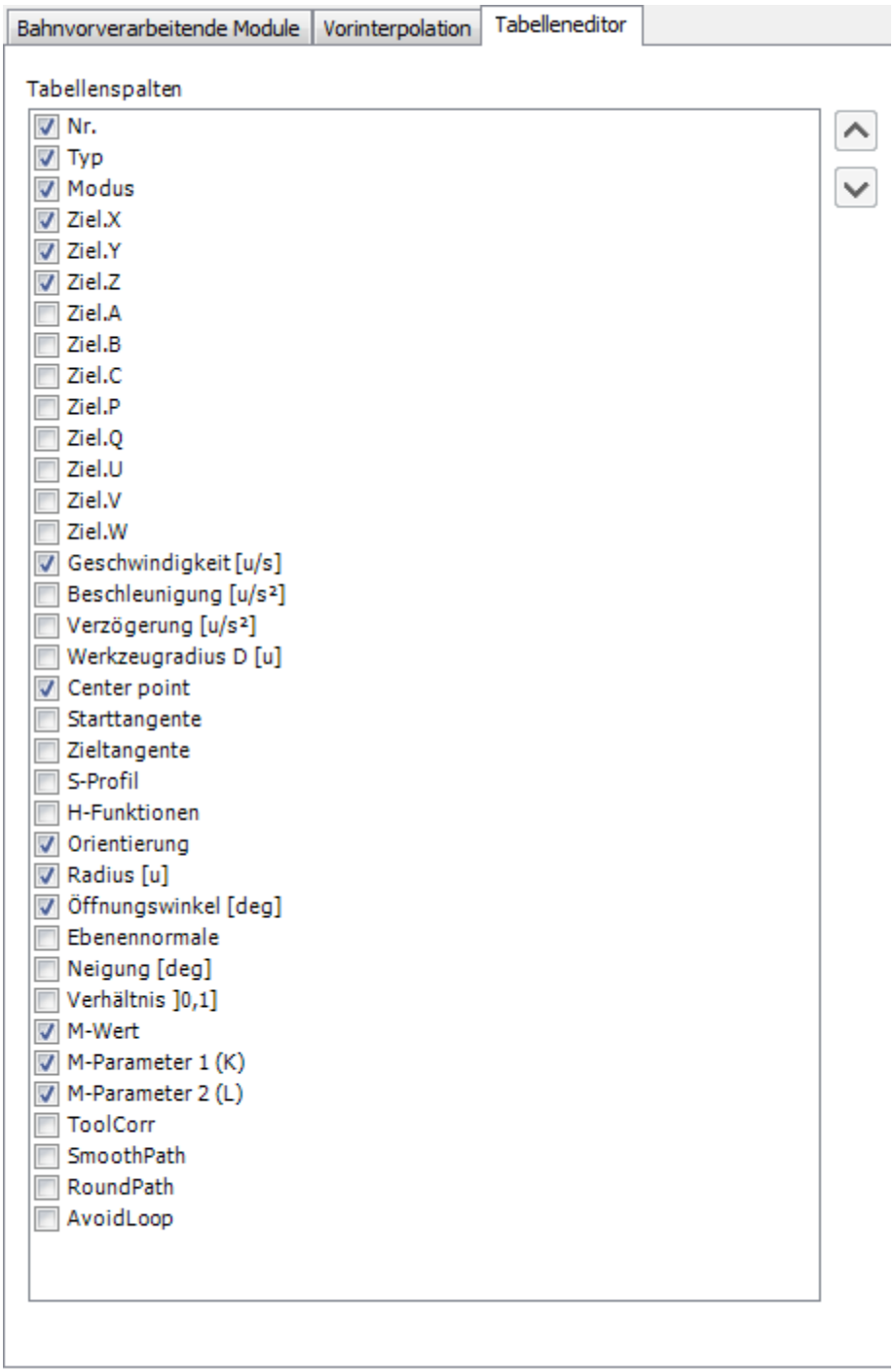


Abb. 347 CNC-Einstellungen für Tabelleneditor

Hier können die Spalten und deren Anordnung im Tabelleneditor festgelegt werden. In der Liste **Tabellenspalten** sind alle verfügbaren Attribute aufgelistet. Nur diejenigen Attribute, für die das Häkchen gesetzt ist, sind im Tabelleneditor sichtbar. Das Häkchen kann durch einen Klick auf die Checkbox oder durch Drücken der **Leertaste** gesetzt oder weggenommen werden.

Um die Reihenfolge der Attribute festzulegen, ist oder **Umschalttaste + Nach-Oben-Cursortaste**für das Verschieben nach oben und oder **Umschalttaste + Nach-Unten-Cursortaste**für das Verschieben nach unten der Attribute innerhalb der Liste zu verwenden. Die Tabellenspalte Typ kann nicht auf unsichtbar gesetzt werden. Die Änderungen betreffen sofort alle offenen Tabelleneditoren.

13.4.1 Parameter-Editor

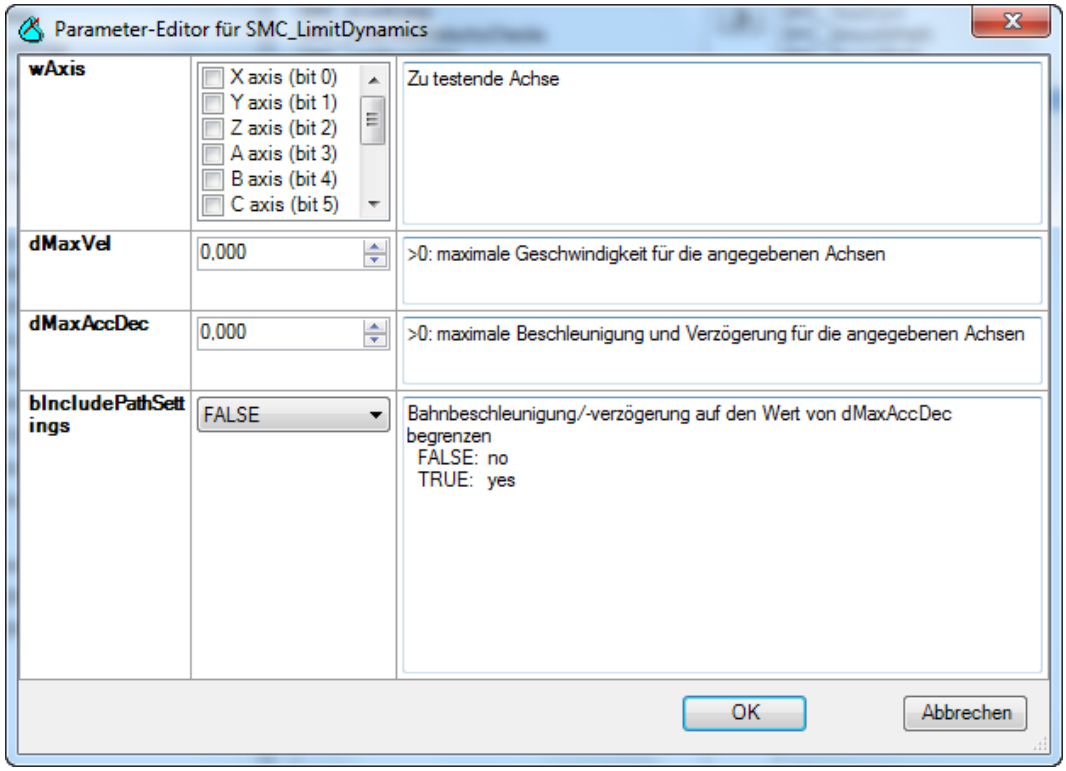


Abb. 348 Dialog Parameter-Editor

In diesem Dialog ist es möglich, die Eingänge der Instanzen von Funktionsblöcken der Bahnvorverarbeitung zu editieren. Bitte beachten sie, dass nicht alle Eingänge sichtbar sind. (Die editierbaren sind mit einem Initialwert vorbelegt.) Die erste Spalte enthält den Namen des Eingangs, die zweite Spalte den änderbaren Wert und kann mit Hilfe des Pulldown-Menüs oder von Checkboxen geändert werden. In der letzten Spalte kann der Kommentar zum Eingang gelesen werden.

13.5 Datenquellen-Editor

13.5.1 Data Server, Überblick

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set verwendet wird.

Ein "Data Server" kann einer Applikation hinzugefügt werden, um eine Punkt-zu-Punkt-Verbindung zwischen dieser Applikation (auf dem "lokalen" Gerät) und einer entfernten Datenquelle einzurichten. Dadurch können **Daten**, d.h. Variablen, die beispielsweise von anderen Geräten oder (derzeit noch nicht möglich) von OPC-Servern bereitgestellt werden, in der lokalen Applikation visualisiert werden. Dabei können einem Data Server mehrere verschiedene Datenquellen zugeordnet werden.

Hinweis: Der Data Server ersetzt nicht die Netzwerkvariablen-Funktionalität, die bereits in »PLC Designer« V2.3 verwendet wurde. Er ermöglicht nur einen anderen Typ des Datenaustauschs. Netzwerkvariablen werden, basierend auf definierten Variablenlisten, im Broadcast-Modus ausgetauscht.

Achtung: In Visualisierungen verwendete Bitzugriffe, die über eine Data Server Verbindung ausgeführt werden, funktionieren nur, wenn sie literale Offsetangaben (also keine definierten Konstanten) enthalten.

Ein **Data Server** muss im Geräte-Fenster als Objekt unterhalb einer Applikation eingefügt werden. Nur 1 Data Server per Applikation kann eingerichtet werden. Die erforderlichen Data Server Bibliotheken werden automatisch im Projekt eingebunden.

Die **Datenquelle(n)**, die der Data Server handhaben soll, müssen als Objekte unterhalb des Data Servers eingefügt werden.

Der Datenquellen-Editor bietet Dialoge zur Konfiguration der einzelnen Datenquellen. Konfiguriert wird zum einen, welche Variablen berücksichtigt werden sollen, welche Zugriffe auf diese möglich sein sollen und zum anderen die Einstellungen für die Kommunikation mit der Datenquelle.

Die Funktionalität Data Server / Datenquelle wird durch die Plug-Ins 'DataSourceEditor' und 'DataSourceObject' bereitgestellt.

13.5.2 Data Server hinzufügen

Ein Data Server Objekt kann via Befehl 'Objekt hinzufügen' einer Applikation hinzugefügt werden. Pro Applikation kann nur 1 Data Server aufgesetzt werden und das Objekt kann nicht umbenannt werden.

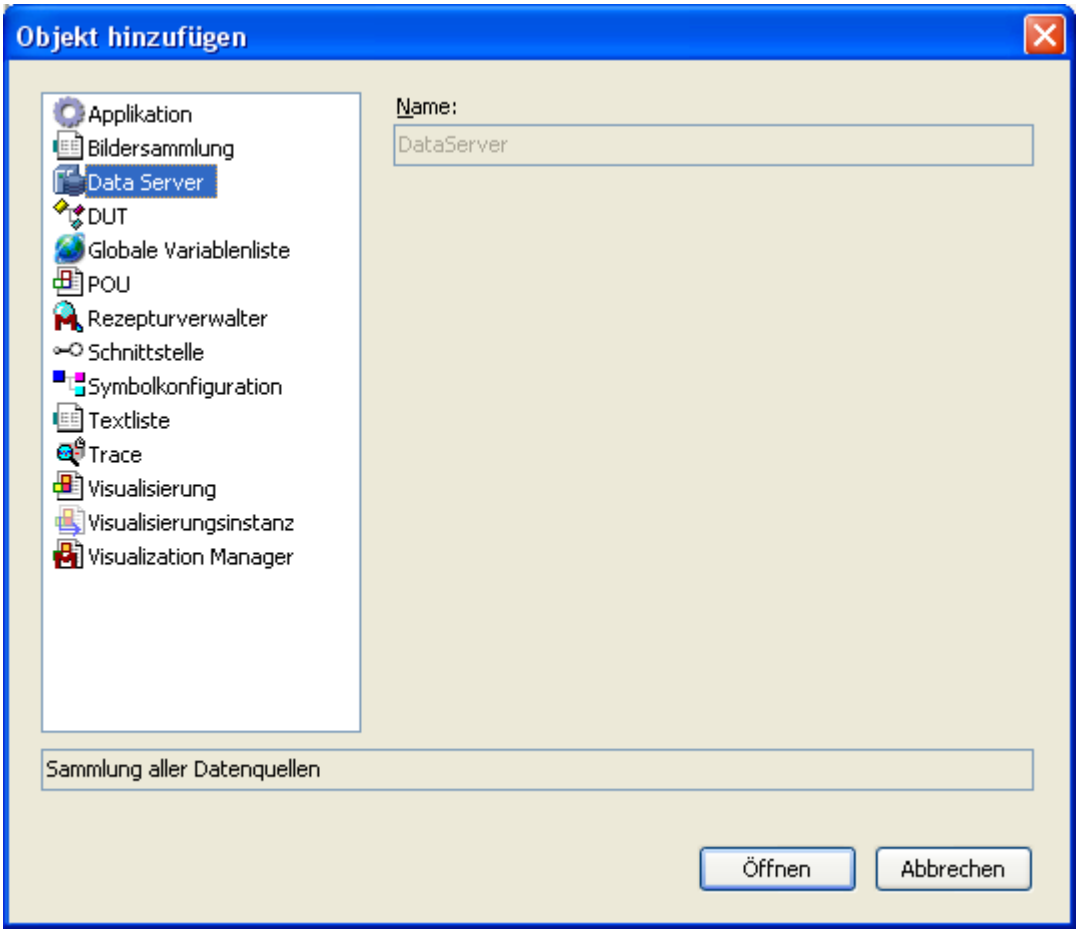


Abb. 349 Dialog 'Objekt hinzufügen', Data Server hinzufügen

Mit dem Data Server Objekt werden automatisch die erforderlichen Data Server Bibliotheken im projektglobalen Bibliotheksverwalter eingefügt und eine Data Server Task wird in der Taskkonfiguration der Applikation angelegt.

Unterhalb des Data Servers können mit dem Befehl 'Objekt hinzufügen' (Datenquelle) eine oder mehrere **Datenquellen**-Objekte eingehängt werden.

13.5.3 Datenquelle hinzufügen

Eine Datenquelle kann dem Data Server einer Applikation via Befehl 'Objekt hinzufügen' zugewiesen werden.

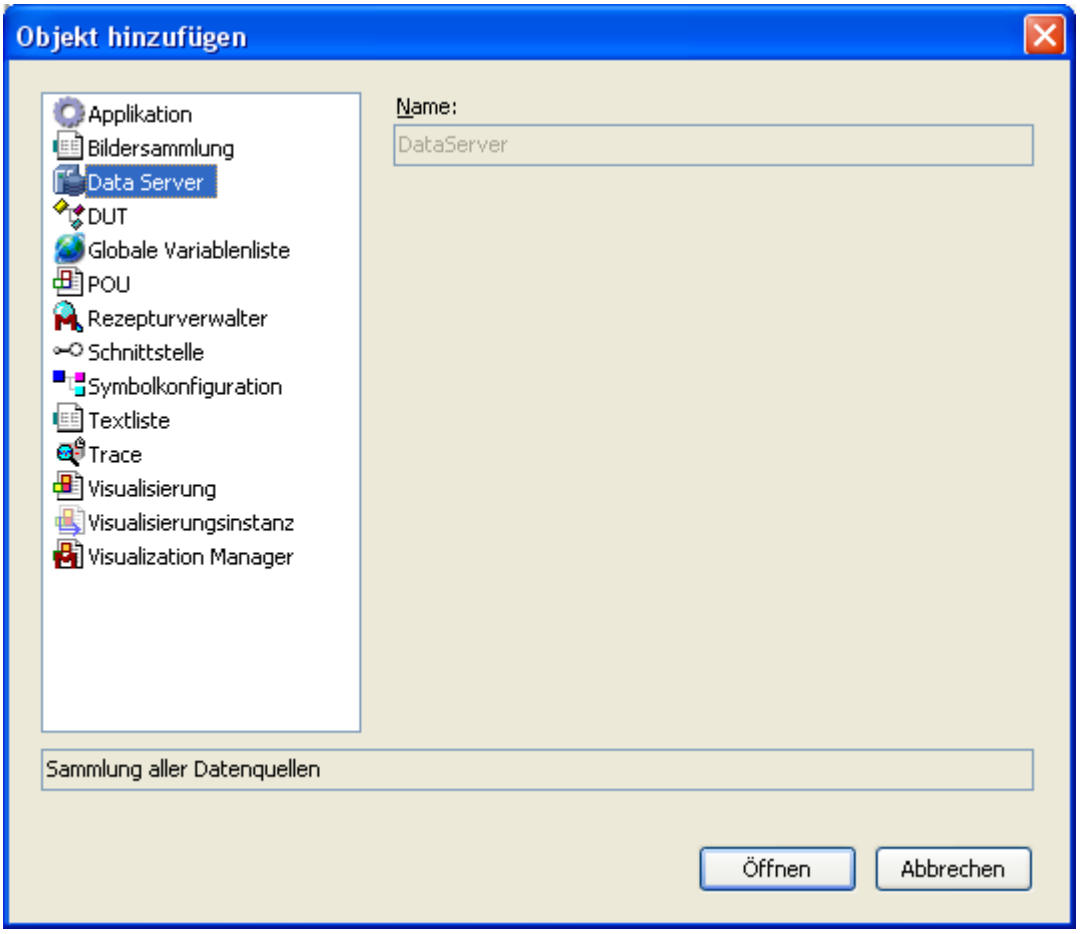


Abb. 350 Dialog 'Objekt hinzufügen', Datenquelle hinzufügen

Der Dialog 'Objekt hinzufügen' bietet eine Auswahlliste (**Typ der Datenquelle:**), über die der gewünschte Typ Datenquelle eingestellt werden muss. Derzeit verfügbar: "Standard SPS" für 3S Geräte; noch nicht implementiert: OPC Server.

Im Fenster unterhalb der Auswahlliste werden dann die aktuell verfügbaren Datenquellen des eingestellten Typs angezeigt. Wählen Sie eine Datenquelle aus und vergeben für sie einen symbolischen **Name**n

Nach Bestätigen mit **Öffnen** wird das Datenquellen-Objekt mit dem symbolischen Namen im Gerätebaum unter dem Data Server der Applikation eingehängt.

Beispiel:

Der Gerätebaum enthält aktuell drei Geräte-Einträge: Device1, Device2, MyPlc.

Applikation App1 unterhalb von MyPlc möchte die Daten von Gerät Dev_186 und von Gerät Dev_PPC verwenden. Um dies zu erreichen, fügen Sie unterhalb App1 erst ein Data Server Objekt ein und darunter ein Datenquellen-Objekt. Im 'Objekt hinzufügen'-Dialog wählen Sie dabei für die Datenquelle Typ '3S SPS". Daraufhin werden Dev_186/Applikation<xy> und Dev_PPC/Applikation<xy> angeboten. Wählen Sie erst die Applikation von Dev_186 und geben den Namen "DataSource_Dev_186" ein. Bestätigen Sie mit 'Öffnen'. Die Datenquelle wird im Gerätebaum eingefügt. Wiederholen Sie das ganze, um als zweite Datenquelle die Applikation von Dev_PPC einzufügen, diesmal mit Namen "DataSource_Dev_PPC". Nun sollten beide Quellen im Gerätebaum eingehängt sein, wie in der untenstehenden Abbildung gezeigt, und können nun konfiguriert werden.

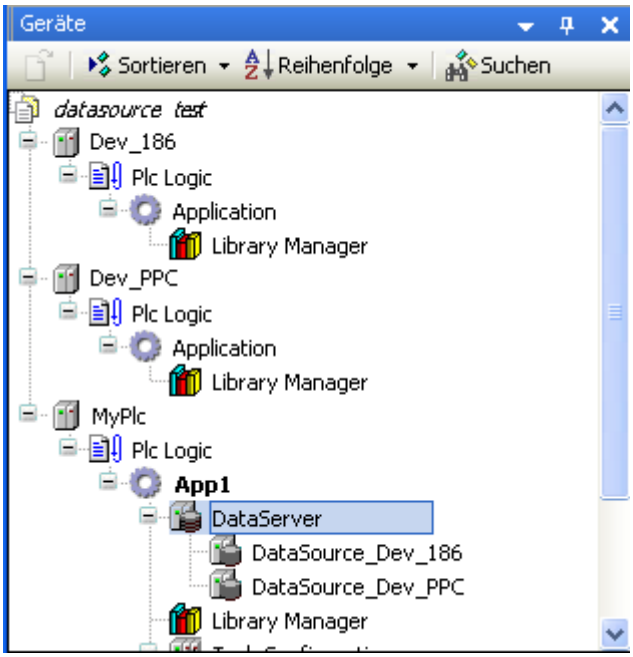


Abb. 351 Beispiel, Data Server und Datenquellen

13.5.4 Datenquelle konfigurieren

Die Datenquellen-Objekte, die einem Data Server zugewiesen wurden, können bezüglich der Auswahl der Variablen und der Kommunikation mit der tatsächlichen Datenquelle konfiguriert werden:

Öffnen Sie den Konfigurationsdialog mit Befehl 'Objekt bearbeiten' oder einem Doppelklick auf das Datenquellen-Objekt im Gerätebaum.

Der Dialog hat zwei Registerblätter: Datenquellenvariablen, Kommunikation.

Datenquellenvariablen

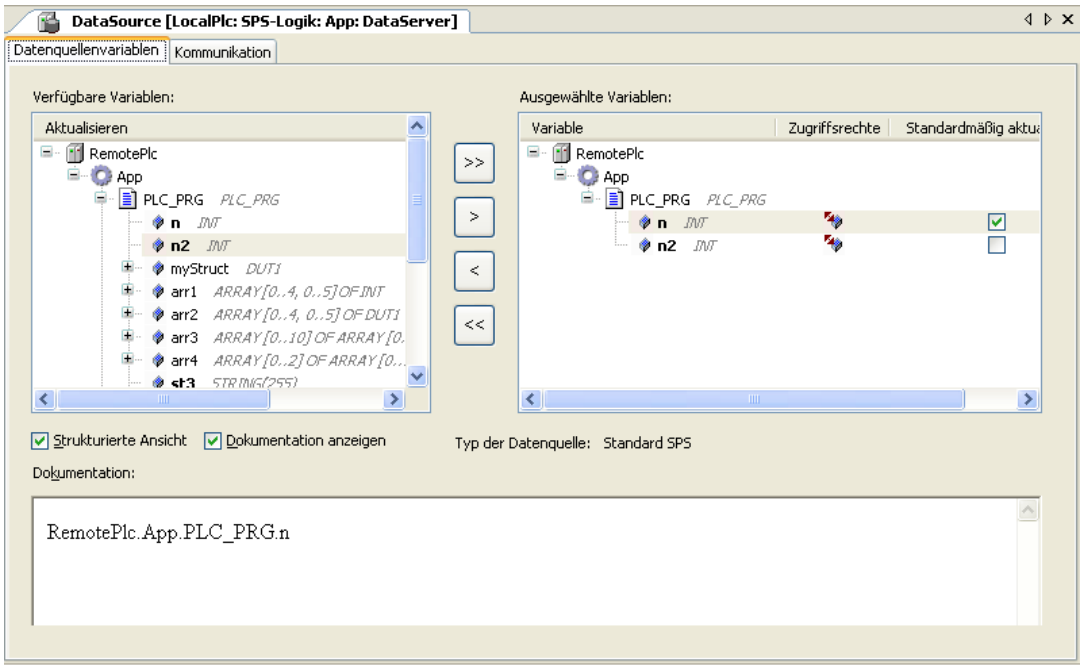


Abb. 352 Dialog 'DataSource', Datenquellenvariablen

Hier wird festgelegt, welche Variablen aus der entfernten Datenquelle in der lokalen Applikation verwendet werden können und welche Zugriffsrechte dabei gelten sollen.

Hinweis: In Visualisierungen verwendete Bitzugriffe, die über eine Data Server Verbindung ausgeführt werden, funktionieren nur, wenn sie literale Offsetangaben (also keine definierten Konstanten) enthalten.

Alle Variablen der Datenquelle werden im Fenster **Verfügbare Variablen** angezeigt. Mit Schaltfläche **Aktualisieren** oberhalb der Variablenliste kann die Anzeige aktualisiert werden. Damit die Liste der Variablen erstellt werden kann, muss für die zugehörige Applikation zuvor Code erzeugt worden sein!

Wenn die Option **Strukturierte Ansicht** aktiviert ist, werden die Variablen in Baumstruktur dargestellt, ansonsten nicht hierarchisch in "flacher" Anordnung. Im Knoten 'Applikation' erscheinen die von der Datenquelle-Applikation verwendeten Bausteine und ihre Variablen. Wenn dabei Funktionsbausteine und Datentypen instanziert werden, werden diese und ihre Komponenten unterhalb des Knotens 'Datentypen' dargestellt. Achtung, bei flacher Anordnung fehlt die Darstellung der Datentypen-Bausteine!

Wenn die Option **Dokumentation anzeigen** aktiviert ist, erscheint im Feld **Dokumentation**, soweit von der Datenquelle bereitgestellt, eine Beschreibung der gerade oben ausgewählten Variable.

Zur Auswahl der Variablen, auf die die lokale Applikation zugreifen können soll:

Grundsätzlich werden die Variablen im Fenster 'Verfügbare Variablen' ausgewählt und entweder durch einen Doppelklick oder über die Schaltfläche > (einzeln) oder Schaltfläche >> (alle) ins Fenster 'Ausgewählte Variablen' übertragen. Umgekehrt können Variablen mit Hilfe der < und << Schaltflächen auch wieder von der Auswahlliste entfernt werden.

In der **strukturierten** Ansicht gibt es die Möglichkeit, die **Auswahl** im Falle von **Funktionsbausteinen** und **Datentypen** auf bestimmte Bausteinkomponenten **einzuschränken**. Gehen Sie folgendermaßen vor:

Wählen Sie im Fenster 'Verfügbare Variablen' im Knoten 'Applikation' die gewünschten Variablen aus. Wenn Instanzen von Funktionsbausteinen und Datentypen dabei sind, werden automatisch alle deren Komponenten bereitgestellt. Wenn dies nicht gewollt ist, kann die Veröffentlichung auf bestimmte Komponenten beschränkt werden, indem diese in Knoten 'Datentypen' explizit ausgewählt werden.

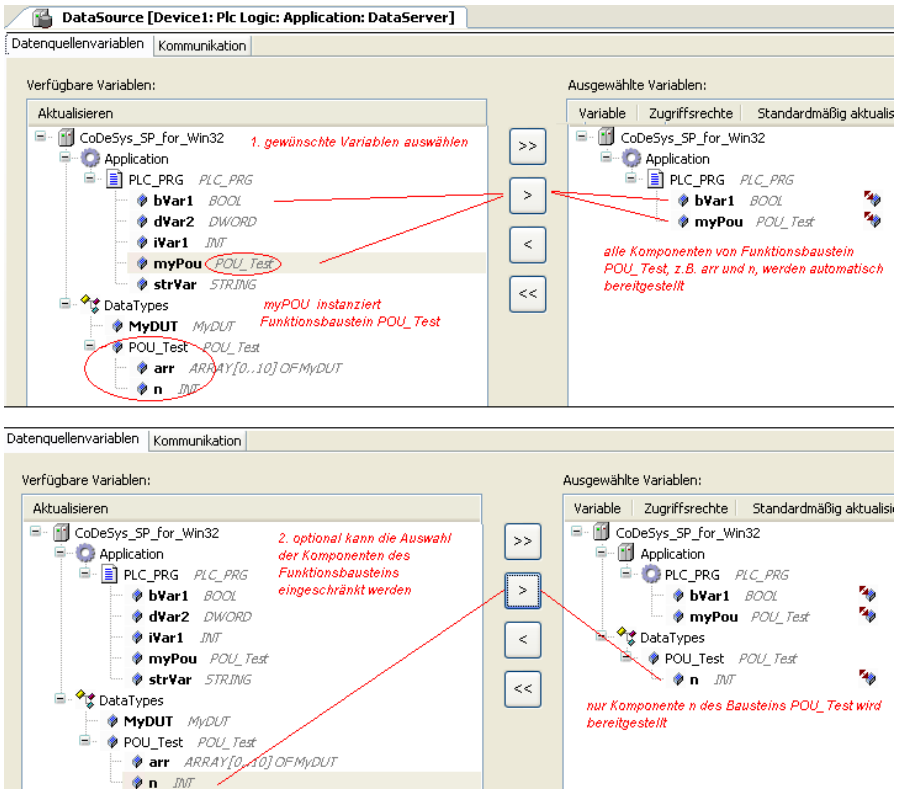


Abb. 353 Beispiel

Die konfigurierte Auswahl von Variablen wird **unmittelbar wirksam**, d.h. die Applikation kann die Variablen sofort verwenden.

Zugriffsrechte:

Standardmäßig wird zunächst für jede Variable das Zugriffsrecht 'lesen und schreiben' vergeben (siehe Spalte **Zugriffsrechte**). Um ein Zugriffsrecht in 'Nur lesbar' zu ändern, klicken Sie auf das Symbol in dieser Spalte um es anzuwählen und ein weiteres Mal, um es "umzuschalten".

- Symbol 'lesen und schreiben':
- Symbol 'nur lesen':

Folgende Bedeutung hat das für eine Variable gesetzte Zugriffsrecht, wenn zusätzlich die Option **Standardmäßig aktualisieren** aktiviert ist (durch Setzen eines Hakens in der gleichnamigen Spalte) :

lesen: Die Variable wird in der lokalen Applikation mit dem Wert, den sie gerade in der Datenquelle hat, aktualisiert, sobald sich dieser in der Datenquelle geändert hat.

schreiben: Die Variable wird in der Datenquelle mit dem Wert, den sie gerade in der lokalen Applikation hat, aktualisiert, sobald sich dieser in der Applikation geändert hat.

Hinweis: Standardmäßig werden die ausgewählten Variablen immer nur in der gerade aktiven Visualisierung aktualisiert. Wenn eine Variable auch in einem anderen Baustein der Applikation verwendet wird, muss sie dort explizit durch die entsprechenden Funktionen des Data Server Interfaces aktualisiert werden! (Wenden Sie sich in diesem Fall bitte derzeit an Ihren Software-Lieferanten).

Kommunikation

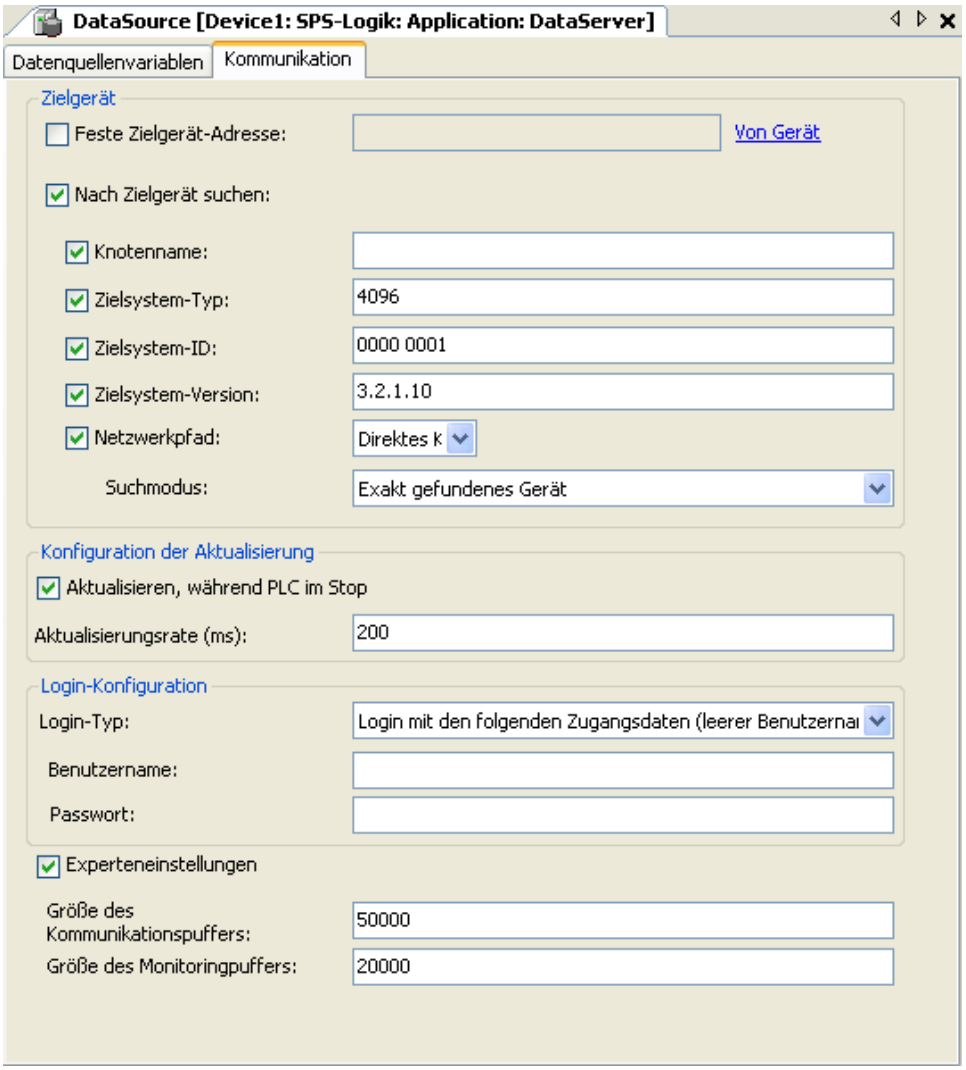


Abb. 354 Dialog 'DataSource', Kommunikationsparameter

Hier werden die Parameter für die Kommunikation mit der Datenquelle auf einem anderen Gerät konfiguriert. Beachten Sie den oben stehenden Hinweis!

1. Datenquellengerät angeben:

Entweder wird eine feste Adresse vorgegeben oder Kriterien für eine dann jeweils aktuelle Suche nach einer verfügbaren Steuerung.

Zielgerät

Durch einen Mausklick auf '**Vom Gerät holen**' werden jeweils die Daten des aktuell angebundenen Datenquellengeräts automatisch eingetragen, so wie sie in deren Kommunikationseinstellungen konfiguriert sind.

Feste Adresse der verknüpften Steuerung: Wenn diese Option aktiviert ist, findet die Kommunikation über die hier angegeben Adresse statt. Geben Sie also die Adresse des Geräts ein, auf dem die Datenquelle liegt, entweder manuell oder über 'Vom Gerät holen' ein.

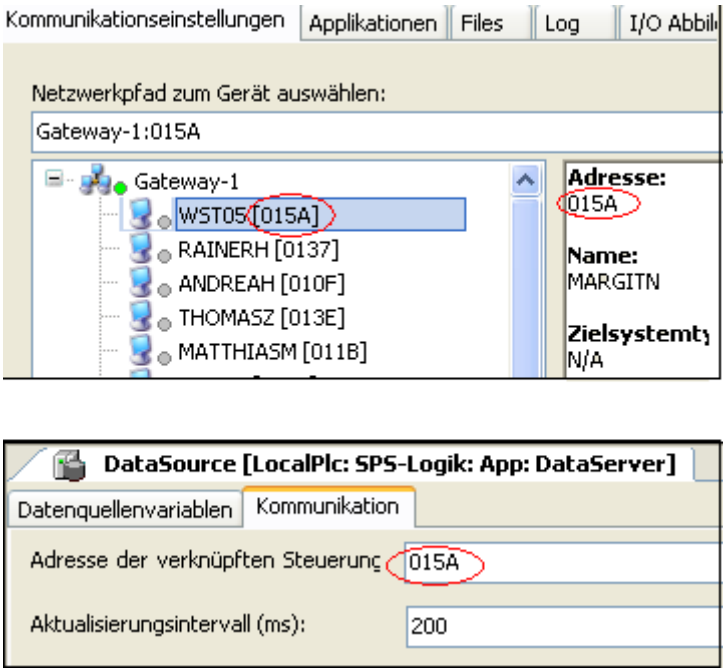


Abb. 355 Beispiel für das Eintragen der Adresse des Datenquellen-Geräts.

Nach Zielgerät suchen: Als Alternative zur Verwendung einer festen Adresse (s.o.) kann aktuell nach einer Steuerung gesucht werden. Dazu können bestimmte Suchkriterien definiert werden (Setzen eines Hakens in die jeweilige Optionenbox):

Knotenname: z.B. WST06

Zielsystem-Typ: Typennummer der Steuerung, z.B. "4096"

Zielsystem-Id: z.B. "0000 0001"

Zielsystem-Version: z.B. "1.0.0.0"

Netzwerkposition: Wählen Sie aus der Auswahlliste, wo sich das Zielgerät befindet:

Direktes Kind der Dataserver-SPS: In diesem Fall muss das Laufzeitsystem, welches den Dataserver betreibt, einen Vermittlungsknoten bereitstellen, auf den IEC zugreifen kann. In der Gerätebeschreibungsdatei des Geräts, auf dem sich der Dataserver befindet, muss zusätzlich die Präprozessorkonstante CMPROUTER_AVAILABLE gesetzt sein um eine mögliche Benutzung der Router-Bibliothek zu signalisieren.

Direktes Kind des Knotens mit Adresse: Bei Auswahl dieser Option geben Sie bitte im zusätzlich erscheinenden Feld die Adresse des Vaterknotens an.

Direktes Kind der Dataserver-SPS oder des Knotens mit Adresse: Diese Auswahl vereint die beiden ersten Optionen. Geben Sie bitte im zusätzlich erscheinenden Feld die Adresse des Vaterknotens an und beachten Sie die hierfür notwendigen Anforderungen an den Router (sh. erste Option).

Suchmodus: Wählen sie aus der Auswahlliste, wie die Suche ablaufen soll:

Erstes gefundenes Gerät: Die erste Steuerung im Gerätebaum, die die angegeben Kriterien erfüllt, wird verwendet.

Exakt gefundenes Gerät: Nur die Steuerung, die genau die angegebenen Kriterien erfüllt, wird verwendet.

2. Setzen der Übertragungsparameter:

Login-Konfiguration

Login-Konfiguration: Wählen Sie von der Liste eine der folgenden Optionen, wie das Einloggen auf das Zielgerät gehandhabt werden soll bzw. welche Zugangsdaten erforderlich sein sollen:

- Login mit den folgenden Zugangsdaten (leerer Benutzername/Passwort ok, wenn keine Benutzerverwaltung auf der SPS): Die Zugangsdaten, die vom Zielgerät gefordert werden, müssen in die folgenden Felder **Benutzername** und **Passwort** eingetragen werden. Wenn das Zielgerät keine Benutzerverwaltung hat, können die Felder leer bleiben.
- Login mit den Zugangsdaten, die zur Laufzeit bestimmt werden: Diese Option sollte gewählt werden, wenn der Benutzer zur Laufzeit über einen Login-Dialogs in einer Visualisierung aufgefordert wird, die gültigen Zugangsdaten einzugeben.
- Keinen Login auf dem Gerät durchführen: Diese Option sollte gewählt werden, wenn ein altes Zielgerät ohne Benutzerverwaltung verwendet wird, damit die Login-Prozedur ganz entfällt.

Konfiguration der Aktualisierung

Aktualisieren, während PLC im Stop: Wenn diese Option aktiviert ist, werden die Datenquellen-Variablen im eingestellten Aktualisierungsintervall auch dann aktualisiert, wenn die Steuerung im Stop ist. Andernfalls wird im Stop-Status nicht aktualisiert.

Aktualisierungsintervall (ms): Aktualisierungsintervall für die Variablen in Millisekunden. Standardwert: 200 ms.

Experteneinstellungen: Wenn diese Option aktiviert ist, wird der Dialog um folgende Einstellungen erweitert:

Größe des Kommunikationspuffers: Puffergröße für Kommunikationsdaten in Bytes. Standardwert: 50000.

Größe des Monitoringpuffers: Puffergröße für die Monitoring-Daten in Bytes. Standardwert: 20000.

13.5.5 Datenquellen in Visualisierungen verwenden

Alle Visualisierungen, die einer Applikation zugeordnet sind, können alle Datenquellen verwenden, die ebenfalls dieser Applikation zugeordnet sind.

Die Variablen der Datenquelle, die für den Datenaustausch konfiguriert sind, können bei der Konfiguration der Eigenschaften eines Visualisierungselements verwendet werden. Sie können via Eingabehilfe oder "Komponenten Auflisten"-Funktion, oder manuell eingetragen werden. Dabei muss der komplette Variablenpfad gemäß folgender Syntax angegeben werden: <Gerätename der Datenquelle>.<Bausteinname>.<Variablenname>.

Beispiel:

Sehen Sie unten den Eigenschaften-Dialog für ein Visualisierungselement. Als Textvariable wird Variable "iCount" von einer definierten Datenquelle verwendet. Diese Datenquelle ist Gerät "Device2", und dort die Applikation "App1" mit Baustein "prg1".

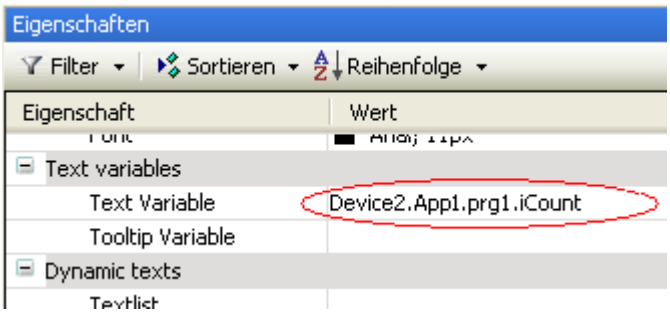


Abb. 356 Beispiel für Verwendung einer Datenquelle-Variablen in den Eigenschaften eines Visualisierungselements

Hinweis: In Visualisierungen verwendete Bitzugriffe, die über eine Data Server Verbindung ausgeführt werden, funktionieren nur, wenn sie literale Offsetangaben (also keine definierten Konstanten) enthalten.

Hinweis: Die Diagnose-Visualisieurng - da sie nur dafür vorgesehen ist, die aktuelle Applikation passiv zu betrachten - zeigt die aktuellen Werte von Variablen, die über einen Data Server aus einer anderen Quelle geliefert werden, nicht an! Für diese werden nur die Initialwerte oder zuletzt gelieferten dargestellt.

Hinweis: Standardmäßig werden die ausgewählten Variablen immer nur in der gerade aktiven Visualisierung aktualisiert. Wenn eine Variable auch in einem anderen Baustein der Applikation verwendet wird, muss sie dort explizit durch die entsprechenden Funktionen des Data Server Interfaces aktualisiert werden! (Wenden Sie sich in diesem Fall bitte derzeit an Ihren Software-Lieferanten).

13.6 Deklarationseditor

Diesen **Texteditor** liefert das Plug-In 'Variable Declaration Object' für die Erstellung des Deklarationsteils eines POU Objekts.

MIt dem 'TabularDeclarationEditor' Plug-in kann er um eine **tabellarische** Ansicht erweitert werden.

Abhängig von den Einstellungen in den Deklarationseditor-Optionen erscheint entweder nur die textuelle oder nur die tabellarische Ansicht, oder es es kann über Schaltflächen (**Textuell** / **Tabellarisch**) am rechten Rand des Editorfensters zwischen den beiden Ansichten gewechselt werden.

Üblicherweise wird der Deklarationseditor in Verbindung mit einem Programmiersprachen-Editor verwendet, d.h. er erscheint im oberen Teil des Fensters, das geöffnet wird, wenn ein Objekt im Offline- oder Online-Modus bearbeitet bzw. angezeigt wird. Der **Deklarationkopf** beschreibt den POU-Typ (z.B. PROGRAM, FUNCTION_BLOCK, FUNCTION ...) und kann durch POU-globale Pragma-Attribute erweitert sein.

Im Online-Modus ist der Deklarationseditor wie eine 'Überwachen'-Ansicht strukturiert.

Variablendeklaration kann auch in einer Globalen Variablenliste erfolgen und in Datentypen-Objekten (DUTs), die allerdings mit eigenen Editoren arbeiten.

Sehen Sie auch die allgemeinen Informationen zur Variablendeklaration.

Textueller Deklarationseditor

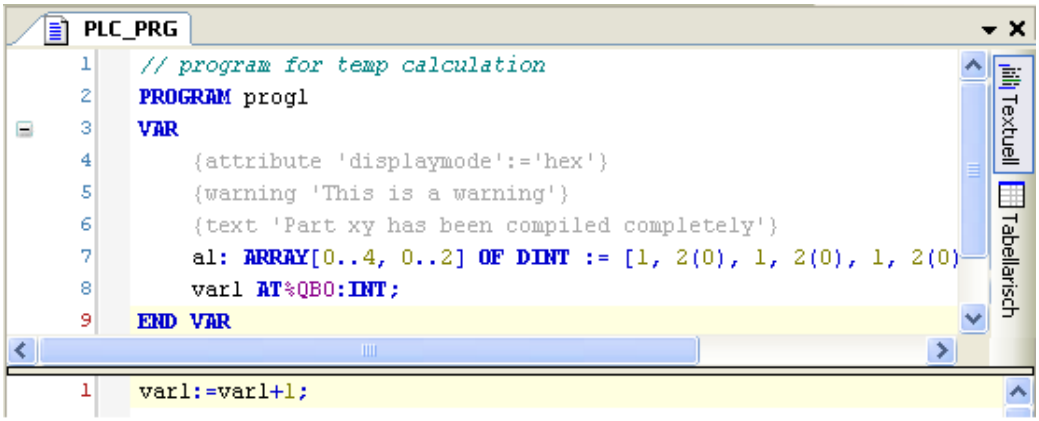


Abb. 357 Textuelle Editoransicht

Verhalten und Aussehen des textuellen Editors sind durch die aktuellen Texteditor-**Einstellungen** in den Optionen- und Anpassen-Dialogen bestimmt. Dort können die Standardeinstellungen für Farben, Zeilennummern, Tab-Breiten, Einrückungen etc. vorgenommen werden. Die üblichen Windows-Funktionen stehen zur Verfügung, auch diejenigen der IntelliMouse werden ggfs. unterstützt. Beachten Sie, dass **Blockselektion** möglich ist, indem die <Alt>-Taste gedrückt wird, während der gewünschte Textteil mit der Maus ausgewählt wird.

Tabelleneditor

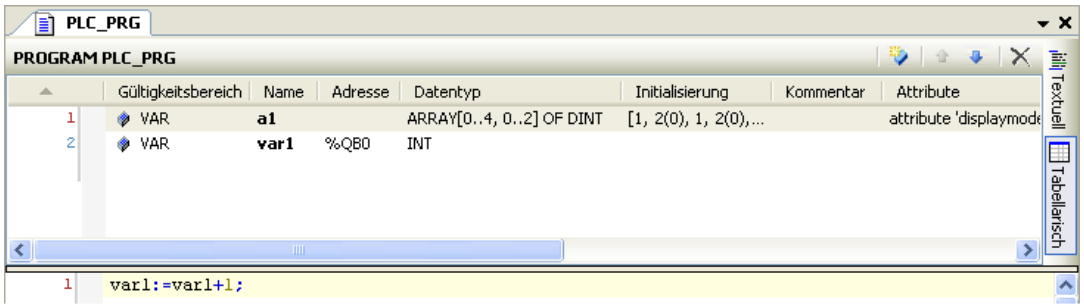


Abb. 358 Tabularische Editoransicht

Die tabellarische Version des Editors zeigt Spalten für die üblichen Definitionen, die für eine Variablendeklaration nötig sind: Gültigkeitsbereich (Sichtbarkeit), Name, Adresse, Datentyp, Initialisierung, Kommentar und (Pragma) Attribute. Die einzelnen Deklarationen werden als nummerierte Zeilen eingefügt.

Der Deklarations-Header kann im Dialog **Deklarationskopf editieren** bearbeitet werden, der mit dem gleichnamigen Befehl aus dem Kontextmenü geöffnet wird.

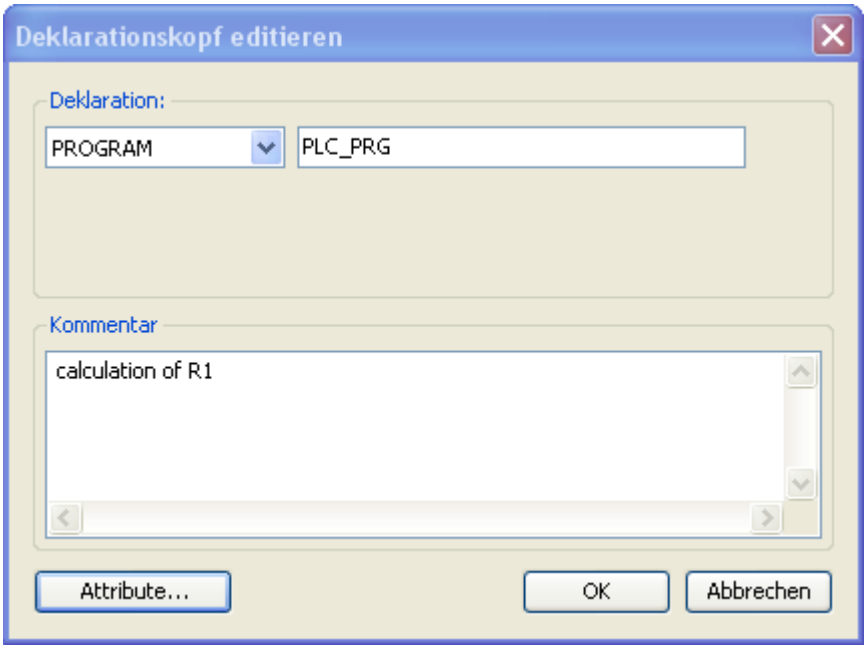


Abb. 359 Dialog Deklarationskopf editieren

Unterhalb "Deklaration:" können Typ (aus der Auswahlliste) und Name des POU Objekts eingegeben werden. Ein Kommentar kann eingetippt werden (Zeilenumbrüche mit Ctrl+Enter) und mit Schaltfläche **Attributes** kann der Attribute Dialog geöffnet werden um Pragmas und Attribute einzugeben.

Um eine **neue Deklarationszeile** oberhalb einer vorhandenen einzufügen, selektiert man die vorhandene und wählt Befehl **Einfügen** aus der Werkzeugleiste oder dem Kontextmenü. Um eine neue Deklaration ganz unten in der Tabelle hinzuzufügen, klickt man auf die letzte bestehende Zeile und verwendet ebenfalls den Einfügen-Befehl.

Dei neu eingefügte Deklaration wird zunächst mit Gültigkeitsbereich "VAR" und dem zuletzt eingegebenen Datentyp versehen. Automatisch öffnet sich das Eingabefeld für den obligatorischen Variablen**Name**n, wo ein gültiger Bezeichner eingegeben werden muss und das dann mit <Eingabetaste> oder einem Mausklick in einen anderen Bereich der Ansicht geschlossen wird.

Jede Tabellenzelle öffnet mit einem **Doppelklick auf die Zelle** die jeweiligen Eingabemöglichkeiten:

Für die Eingabe des **Gültigkeitsbereich**s öffnet sich eine Auswahlliste, aus der der gewünschte Bereich und das passende Bereichsattribut (Flag) gewählt werden können.

Die Eingabe des **Datentyps** kann direkt erfolgen oder man gelangt über Schaltfläche **›** zur Eingabehilfe oder dem Array-Assistenten.

Die Eingabe eines **Initialisierung**swerts erfolgt direkt oder man verwendet über Schaltfläche der Dialog **Initialisierungswert**, der bei strukturierten Variablen hilfreich ist.

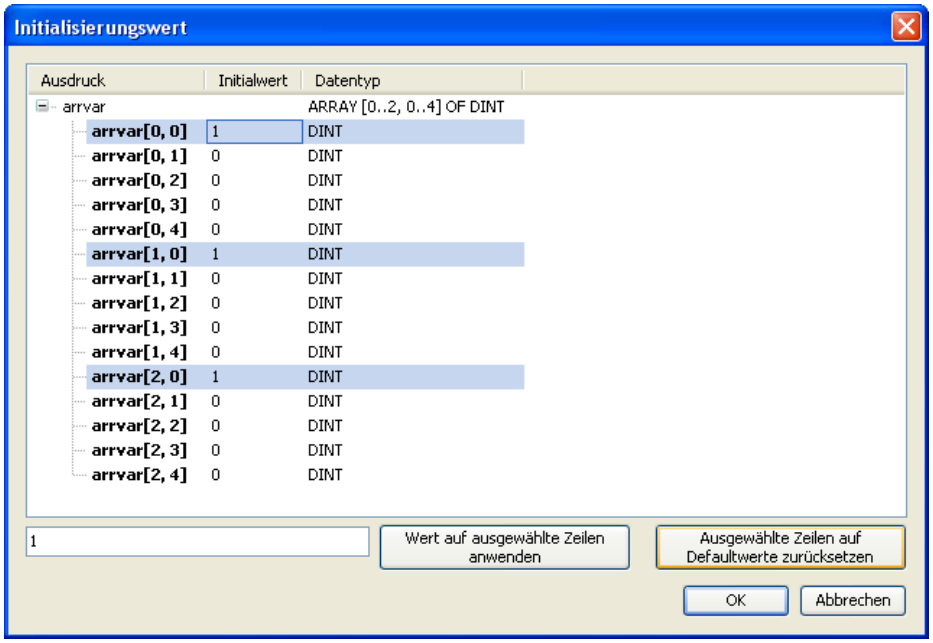


Abb. 360 Initialisierungswert Dialog

Alle **Ausdrücke** der Variablen werden mit ihren aktuellen **Initalwerte**n dargestellt. Wählen Sie die gewünschten aus und bearbeiten den Wert im Feld unterhalb der Liste. Danach wird er über Schaltfläche **Wert auf ausgewählte Zeilen anwenden** übertragen. Über Schaltfläche **Ausgewählte Zeilen auf Defaultwerte zurücksetzen** können die Standard-Initialisierungen wieder hergestellt werden.

Zeilenumbrüche in den Eingabefeldern der **Kommentar**spalte können mit Ctrl+Enter eingefügt werden.

Die **Attribute** Einträge erfolgen im Dialog **Attribute**, wo mehrere Attribute und Pragma Anweisungen im Textformat eingegeben werden können. Sie müssen ohne die {} Klammerung und jeweils in einer eigenen Zeile eingetragen werden. Das in der folgenden Abbildung gezeigte Beispiel entspricht der textuellen Ansicht, die man oben in Abbildung "Textuelle Editoransicht" sieht.

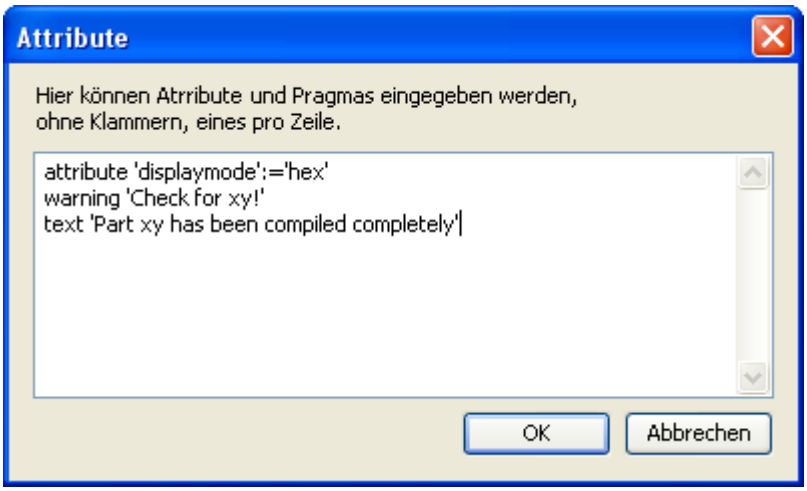


Abb. 361 Attribute Dialog, Beispiel

Jede Variable wird in einer eigenen Zeile deklariert, die **Zeilen sind nummeriert**.

Die **Reihenfolge der Zeilen** (Zeilennummern) kann mit Hilfe der Befehle **Nach oben verschieben** bzw. **Nach unten verschieben** (Kontextmenü oder Werkzeugleiste) verändert werden. Sie verschieben die gerade ausgewählte Zeile um eine Position.

Die Liste der Deklarationen kann nach einer der Spalten **sortiert** werden, indem auf die Spaltenüberschrift geklickt wird. Die Spalte, die gerade die Sortierung bestimmt, wird mit einem Pfeilsymbol gekennzeichnet: (aufsteigend sortiert) oder (absteigend sortiert). Mit jedem weiteren Klick in die Spaltenüberschrift, wird zwischen aufsteigend und absteigend gewechselt.

Zum **Löschen einer oder mehrerer Deklarationen**, wählt man die betreffenden Zeilen aus und verwendet <Entf> oder den Befehl **Löschen** aus Kontextemenü oder Werkzeugleiste ()

Deklarationseditor im Online-Modus

Nach dem Einloggen auf dem Zielsystem wird jedes Projektobjekt, das bereits im Offline-Modus in einem Fenster angezeigt wurde, automatisch in der Online-Ansicht dargestellt.

Die Online-Ansicht des Deklarationseditors zeigt eine Tabelle, wie sie auch in 'Überwachen'-Fenstern verwendet wird. Die Kopfzeile enthält immer den aktuellen Objektpfad "<Gerätename>.<Applikationsname>.<Objektname>". Die Tabelle zeigt für jeden **Ausdruck (watch expression) den Datentyp** und den aktuellen **Wert**, sowie - wenn gerade gesetzt - den zum Forcen oder Schreiben. **Vorbereiteten Wert**.

Zur Vorbereitung eines Werts für eine Variable kann der Dialog Prepare value dialog benutzt werden. Alternativ kann der Wert (nach einem Mausklick) im zugehörigen Feld der Spalte **Vorbereiteter Wert** direkt eingetragen werden.
Im Fall einer booleschen Variable ist die Handhabung sogar noch einfacher: Sie können den vorbereiteten Wert durch Benutzen der **Return-** oder **Leerzeichen-**Taste toggeln, wobei folgende Reihenfolge eingehalten wird: War der vorbereitete Wert TRUE, so folgen die Eingaben FALSE -> TRUE -> leer; ansonsten, bei vorbereitetem Wert FALSE, folgen die Eingaben TRUE -> FALSE -> leer.

Wenn ein Ausdruck (Variable) eine Instanz ist, z.B. eines Funktionsblocks, ist ein Plus- bzw. Minuszeichen vorangestellt. Mit einem Mausklick auf dieses Zeichen werden die einzelnen Komponenten des instanzierten Objekts sichtbar gemacht (siehe fbinst in der nachfolgenden Abbildung) bzw. ausgeblendet (siehe fbinst2). Symbole zeigen an, ob die jeweilige Variable eine Eingabe-, Ausgabe- oder eine "normale" Variable ist.

Wenn der Mauszeiger auf eine Variable im Implementierungsteil zeigt, erscheint ein **Tooltip** mit der Deklaration und dem Kommentar der Variablen. Siehe folgende Abbildung:

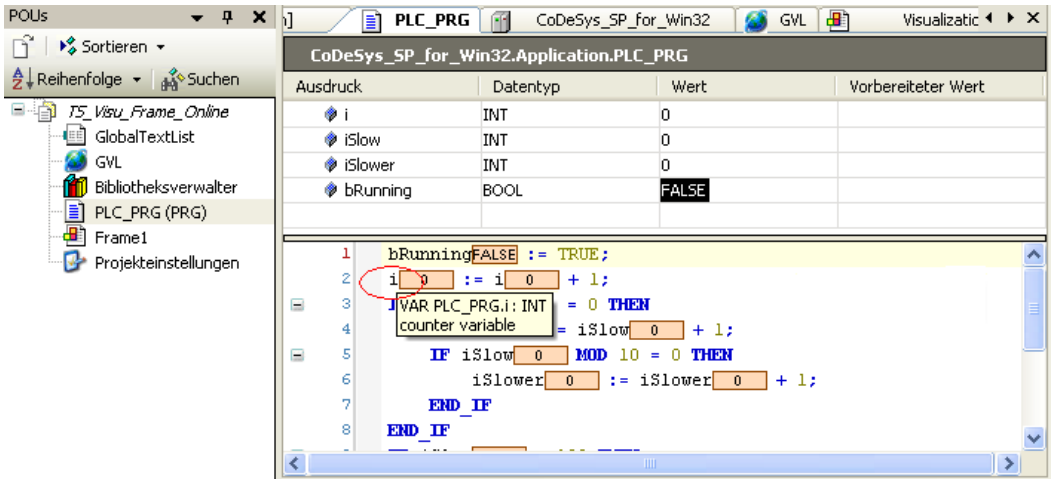


Abb. 362 Beispiel: Deklarationseditor im oberen Teil eines Programm-Objekts PLC_PRG, Online-Ansicht

13.7 DUT-Editor

Benutzerdefinierte Datentypen (DUT, Data Unit Types) können im Datentyp-Editor angelegt werden. Dies ist ein Texteditor und somit werden Aussehen und Verhalten von den aktuellen Einstellungen in den Texteditor-Optionen bestimmt.

Das Fenster des Datentyp-Editors öffnet sich automatisch, sobald ein Datentyp-Objekt (DUT) über den Objekt hinzufügen-Dialog ins Projekt eingefügt wird. In diesem Fall enthält der Editor bereits die Vordefinition einer erweiterten Struktur-Definition, die zu einer einfachen Strukturdefinition oder zur Definition eines anderen Datentyps, z.B. einer Enumeration, abgeändert werden kann.

Der Editor öffnet ebenfalls, wenn ein bestehendes DUT-Objekt zur Bearbeitung geöffnet wird.

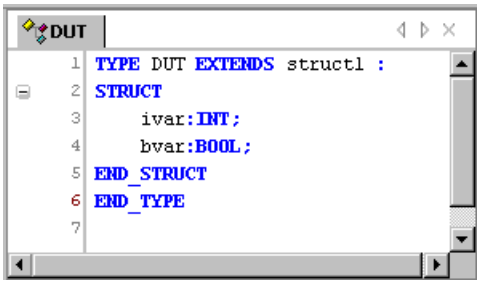


Abb. 363 DUT Editorfenster

13.8 FUP/KOP/AWL-Editor

Für das Programmieren in den Sprachen FUP (Funktionsplan), KOP (Kontaktplan) und AWL (Anweisungsliste) gibt es einen **kombinierten Editor**.

Das bedeutet, dass ein **gemeinsamer Satz an Befehlen und Elementen** verwendet wird und die drei Programmiersprachen automatisch intern ineinander konvertiert werden. Der Programmierer kann im Offline-Modus jederzeit in eine der anderen beiden Editoransichten **umschalten** (Ansicht).

Zu beachten ist allerdings, dass es einige spezielle Elemente gibt, die **nicht konvertiert** werden können und somit nur in der passenden Sprache angezeigt werden. Ebenso gibt es Konstrukte, die zwischen AWL und FUP nicht eindeutig konvertiert werden könnten und deshalb bei der Rückumwandlung in FUP "normalisiert", d.h. wieder aufgehoben werden. Dies betrifft: Negation von Ausdrücken und explizite/implizite Zuweisung von Funktionsblock-Ein- und Ausgängen.

Der FUP/KOP/AWL Editor wird mit dem Plug-In 'Network List Editor' bereitgestellt. **Verhalten, Aussehen und Menüs** werden in den Anpassen- und Optionen-Dialogen definiert. Sehen Sie dort speziell die Einstellmöglichkeiten bezüglich der Darstellung von Kommentaren und Adressen im Editor.

Der Editor wird in einem **zweigeteilten Fenster** geöffnet, wenn ein Objekt, das in FUP/KOP/AWL bearbeitet wird. Der obere Teil enthält den **Deklarationseditor**.

Die **Programmiersprache** für ein neues Objekt wird festgelegt, wenn das Objekt über den 'Objekt hinzufügen'-Dialog angelegt wird.

13.8.1 Programmiersprachen

Funktionsplan - FUP

Der Funktionsplan ist eine grafisch orientierte Programmiersprache. Er arbeitet mit einer Liste von Netzwerken, wobei jedes Netzwerk eine Struktur enthält, die jeweils einen logischen bzw. arithmetischen Ausdruck, den Aufruf eines Funktionsblocks, einen Sprung oder eine Return-Anweisung darstellt.

Der FUP/KOP/AWL Editor wird mit dem Plug-In 'Network List Editor' bereitgestellt.

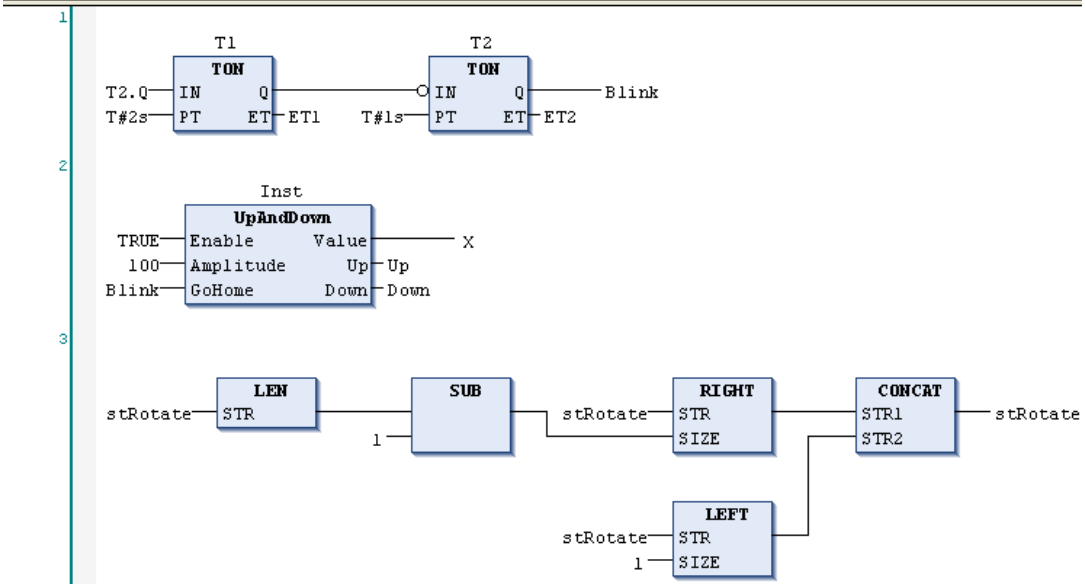


Abb. 364 Beispiel einiger FUP-Netzwerke

Kontaktplan - KOP

Der Kontaktplan ist eine grafisch orientierte Programmiersprache, die dem Prinzip einer elektrischen Schaltung angenähert ist.

Einerseits eignet sich der Kontaktplan dazu, logische Schaltwerke zu konstruieren, andererseits kann man aber auch Netzwerke wie im FUP erstellen. Daher kann der KOP sehr gut dazu benutzt werden, um den Aufruf von anderen Bausteinen zu steuern.

Der Kontaktplan besteht aus einer Folge von Netzwerken. Ein Netzwerk wird auf der linken und rechten Seite von einer linken und einer rechten vertikalen **Stromleitung** begrenzt. Dazwischen befindet sich ein Schaltplan aus Kontakten, Spulen, optional Bausteinen (POUs) und Verbindungslinien.

Jedes Netzwerk besteht auf der linken Seite aus einer Folge von Kontakten, die von links nach rechts den Zustand "AN" oder "AUS" weitergeben, diese Zustände entsprechen den boolschen Werten TRUE und FALSE. Zu jedem Kontakt gehört eine boolsche Variable. Wenn diese Variable TRUE ist, dann wird der Zustand über die Verbindungslinie von links nach rechts weitergegeben, sonst erhält die rechte Verbindung den Wert AUS.

Der FUP/KOP/AWL Editor wird mit dem Plug-In 'Network List Editor' bereitgestellt.

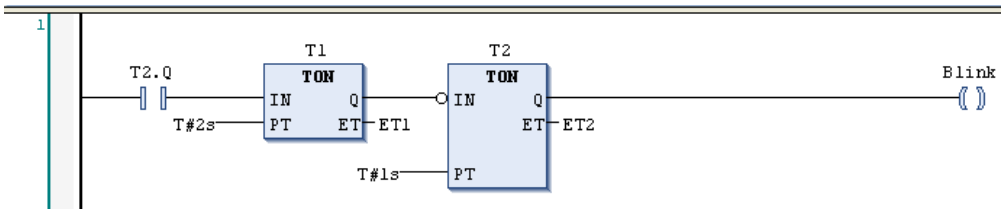


Abb. 365 Beispiel eines KOP-Netzwerks

Anweisungsliste - AWL

Die Anweisungsliste ist eine Assembler-ähnliche IEC61131-konforme Programmiersprache.

Sie unterstützt akkumulator-basiertes Programmieren. Alle IEC 61131-3 Operatoren werden unterstützt, ebenso mehrfache Eingänge, mehrfache Ausgänge, Negationen, Kommentare, Set/Reset von Ausgängen und bedingte/unbedingte Sprünge.

Jede Anweisung basiert primär auf dem Laden von Werten in den Akkumulator (LD-Anweisung). Danach wird die entsprechende Operation mit dem ersten Parameter aus dem Akkumulator ausgeführt. Das Ergebnis der Operation wird wieder in den Akkumulator geschrieben, von wo es der Anwender mit einer ST-Anweisung gezielt speichern sollte.

Für das Programmieren von bedingten Ausführungen oder Schleifen unterstützt die Anweisungsliste Vergleichsoperatoren (EQ, GT, LT, GE, LE, NE) und Sprünge. Sprünge können unbedingt (JMP) oder bedingt (JMPC / JMPCN) sein. Bei bedingten Sprüngen wird geprüft, ob der Wert im Akkumulator TRUE oder FALSE ist.

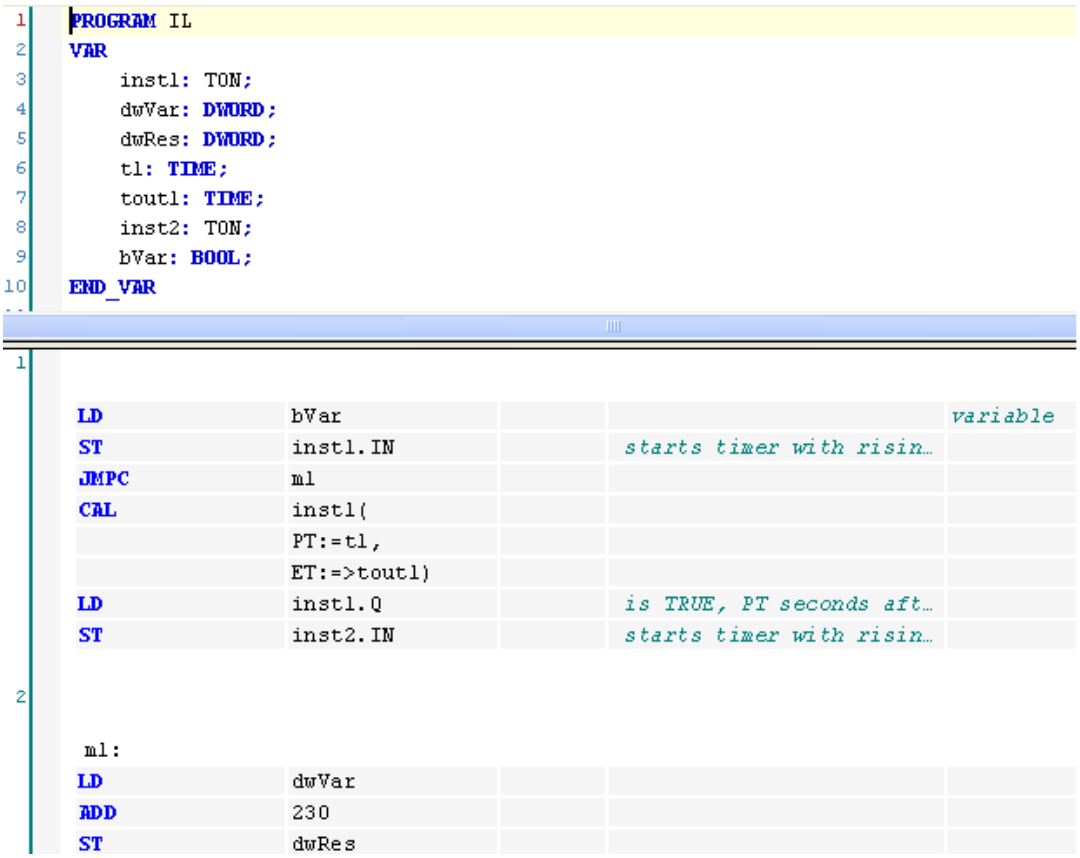
Syntax:

Eine Anweisungsliste (AWL) besteht aus einer Folge von **Anweisungen**. Jede Anweisung beginnt in einer neuen Zeile und beinhaltet einen **Operator** und, je nach Art der Operation, einen oder mehrere durch Kommata separierte **Operanden**.

Vor einer Anweisung kann ein Identifikator (**Marke**) stehen, gefolgt von einem Doppelpunkt (:). Er dient der Kennzeichnung der Anweisung und kann beispielsweise als **Sprungziel** verwendet werden.

Ein **Kommentar** muss das letzte Element in einer Zeile sein. Leere Zeilen können zwischen Anweisungen eingefügt werden

Zwischen Anweisungen können Leerzeilen stehen.



```
PROGRAM IL
VAR
    inst1: TON;
    dwVar: DWORD;
    dwRes: DWORD;
    t1: TIME;
    tout1: TIME;
    inst2: TON;
    bVar: BOOL;
END_VAR
```

```
LD      bVar                                    variable
ST      inst1.IN                starts timer with risin…
JMPC    m1
CAL     inst1(
        PT:=t1,
        ET:=>tout1)
LD      inst1.Q                 is TRUE, PT seconds aft…
ST      inst2.IN                starts timer with risin…

m1:
LD      dwVar
ADD     230
ST      dwRes
```

Abb. 366 Programmbeispiel im AWL-Tabelleneditor

Der AWL-Editor ist ein Tabelleneditor, der im FUP/KOP/AWL Editor integriert ist. Dieser wird mit dem Plug-In 'Network List Editor' bereitgestellt.

13.8.2 Modifikatoren und Operatoren in AWL

In der Sprache Anweisungsliste können folgende Operatoren und Modifikatoren verwendet werden.

Modifikatoren:

C	mit JMP, CAL, RET:	Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks TRUE ist.
N	mit JMPC, CALC, RETC:	Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks FALSE ist.
N	ansonsten:	Negation des Operanden (nicht des Akku).

Im folgenden finden Sie eine Tabelle aller Operatoren in AWL mit deren möglichen Modifikatoren und der jeweiligen Bedeutung:

Der "**Akkumulator**" enthält immer den Ergebniswert der vorangegangenen Operation.

Operator	Modifikatoren	Bedeutung	Beispiel
LD	N	Lädt den (negierten) Wert des Operanden in den Akkumulator	LD iVar
ST	N	Speichert den (negierten) Inhalt des Akkumulators in den Operanden	ST iErg
S		Setzt den Operanden (Typ BOOL) auf TRUE, wenn der Inhalt des Akkumulators TRUE ist.	S bVar1
R		Setzt den Operanden (Typ BOOL) auf FALSE, wenn der Inhalt des Akkumulators TRUE ist.	R bVar1
AND	N,(	Bitweises AND des Akkumulator-Wertes und des (negierten) Operanden	AND bVar2
OR	N,(	Bitweises OR des Akkumulator-Wertes und des (negierten) Operanden	OR xVar
XOR	N,(	Bitweises, exklusives OR des Akkumulator-Wertes und des (negierten) Operanden	XOR N,(bVar1,bVar2)
NOT		Bitweise Negation des Akkumulator-Wertes	
ADD	(	Addition des Akkumulator-Wertes und des Operanden; Ergebnis wird in den Akkumulator geschrieben	ADD (iVar1,iVar2)
SUB	(	Subtraktion des Operanden vom Akkumulator-Wert; Ergebnis wird in den Akkumulator geschrieben	SUB iVar2
MUL	(	Multiplikation von Akkumulator-Wert und Operand; Ergebnis wird in den Akkumulator geschrieben	MUL iVar2
DIV	(	Division des Akkumulator-Wertes durch den Operanden; Ergebnis wird in den Akkumulator geschrieben	DIV 44
GT	(	Prüfen, ob der Akkumulator-Wert größer als der Operandenwert ist; Ergebnis (BOOL) wird in den Akkumulator geschrieben; >	GT 23
GE	(	Prüfen, ob der Akkumulator-Wert größer oder gleich Operandenwert ist; Ergebnis (BOOL) wird in den Akkumulator geschrieben; ; >=	GE iVar2
EQ	(	Prüfen, ob Akkumulator-Wert gleich Operandenwert ist; Ergebnis (BOOL) wird in den Akkumulator geschrieben; ; =	EQ iVar2
NE	(	Prüfen, ob Akkumulator-Wert ungleich Operandenwert ist; Ergebnis (BOOL) wird in den Akkumulator geschrieben; <>	NE iVar1
LE	(	Prüfen, ob Akkumulator-Wert kleiner oder gleich Operandenwert ist; Ergebnis (BOOL) wird in den Akkumulator geschrieben; ; <=	LE 5
LT	(	Prüfen, ob der Akkumulator-Wert kleiner als der	LT cVar1

Operator	Modifikatoren	Bedeutung	Beispiel
		Operandenwert ist; Ergebnis (BOOL) wird in den Akkumulator geschrieben; ; <	
JMP	CN	Unbedingter (bedingter) Sprung zur angegebenen Marke	JMPN next
CAL	CN	(Bedingter) Aufruf eines a Programms oder Funktionsbausteins (wenn der Akkumulator-Wert TRUE ist)	CAL prog1
RET		Verlassen des Bausteins und Rückkehr zum aufrufenden Baustein	RET
RET	C	Bedingtes - wenn der Akkumulator-Wert TRUE ist - Verlassen des Bausteins und Rückkehr zum aufrufenden Baustein	RETC
RET	CN	Bedingtes - wenn der Akkumulator-Wert FALSE ist - Verlassen des Bausteins und Rückkehr zum aufrufenden Baustein	RETCN
)		Auswerten der zurückgestellten Operation	

Siehe hierzu auch: IEC-Operatoren.

Siehe Arbeiten im AWL-Editor zum Gebrauch von **Multiplen Operanden**, von **komplexen Operanden**, von Funktions-/Methoden-/Funktionsblock-/Programm-/Aktions**aufrufen** und von **Sprüngen**.

Das folgende Bild zeigt ein Beispielprogramm, in dem einige Modifikatoren verwendet werden:

```
1
    AND     TRUE    load TRUE to accumulator
    ANDN    bVar1   execute AND with negated value of bVar1
    JMPC    m1      if accum. is TRUE, jump to label "m1"

    LDN     bVar2   store negated value of bVar2...
    ST      bRes    ... in bRes
2
    m1:

    LD      bVar2   store value of bVar2...
    ST      bRes    ... in bRes
```

Abb. 367 Beispiel eines AWL-Programms

13.8.3 Arbeiten im FUP- und KOP-Editor

"**Netzwerke**" sind die Basiseinheiten der FUP- und KOP-Programmierung. Jedes Netzwerk enthält eine Struktur, die einen logischen bzw. arithmetischen Ausdruck, den Aufruf eines Programmierbausteins (Funktion, Funktionsbaustein, Programm etc.), einen Sprung oder eine Return-Anweisung darstellt.

Wenn ein neues Objekt angelegt wird, enthält das Editorfenster automatisch bereits ein leeres Netzwerk.

Beachten Sie die generellen Editor-Einstellungen im Optionen-Dialog, Kategorie 'FUP, KOP und AWL'.

Tooltip mit Informationen zu Variablen oder Bausteinparametern

Wenn der Cursor auf eine Variable oder einen Bausteinparameter zeigt, wird in einem Tooltip der jeweilige Datentyp und bei Funktionsbausteininstanzen der Initialisierungswert dargestellt. Für die IEC Operatoren SEL, LIMIT, MUX erscheint eine kurze Beschreibung der Eingänge. Wenn definiert, werden außerdem die Adresse und der Symbolkommentar dargestellt, sowie in einer zweiten Zeile der Operandenkommentar in doppelten Hochkommas.

Einfügen und Anordnen von Elementen:

- Die Befehle für das Arbeiten im Editor sind standardmäßig im FBD/LD/IL Menü enthalten, die wichtigsten immer auch im Kontextmenü enthalten. Es hängt immer von der aktuellen Cursor-Position bzw. aktuellen Selektion (Mehrfachauswahl möglich) ab, welche Elemente gerade über Menübefehl eingefügt werden können.
- Elemente können auch direkt mit der Maus aus der Werkzeugbox ins Editorfenster gezogen werden, oder von einer Position im Editor zu einer anderen ("Drag&Drop"). Dazu selektiert man das Element mit einem Mausklick, hält dann die Maustaste gedrückt und zieht das Element in das betreffende Netzwerk im Editor. Sobald das Netzwerk erreicht wird, werden alle möglichen Einfügepositionen für den Elementtyp durch graue Positionsmarken angezeigt. Wenn der Mauszeiger auf einer dieser Positionen steht, wechselt die Positionsmarke nach grün und durch Loslassen der Maustaste wird das Element an dieser Position eingefügt.
- Die Befehle Ausschneiden, Kopieren, Einfügen und Löschen, die standardmäßig im Bearbeiten-Menü verfügbar sind, können verwendet werden, um die Elemente anzuordnen. Das Kopieren eines Elements ist auch über Drag&Drop möglich. Man wählt das Element im Netzwerk durch einen Mausklick aus und zieht es bei gedrückt gehaltener Maustaste zur Zielposition. Wenn diese erreicht ist (grüne Positionsmarke), erscheint ein Plus-Symbol am Cursor. Nun läßt man die Maustaste los um das Element einzufügen.
- Siehe auch: Cursor-Positionen in FUP, KOP und AWL

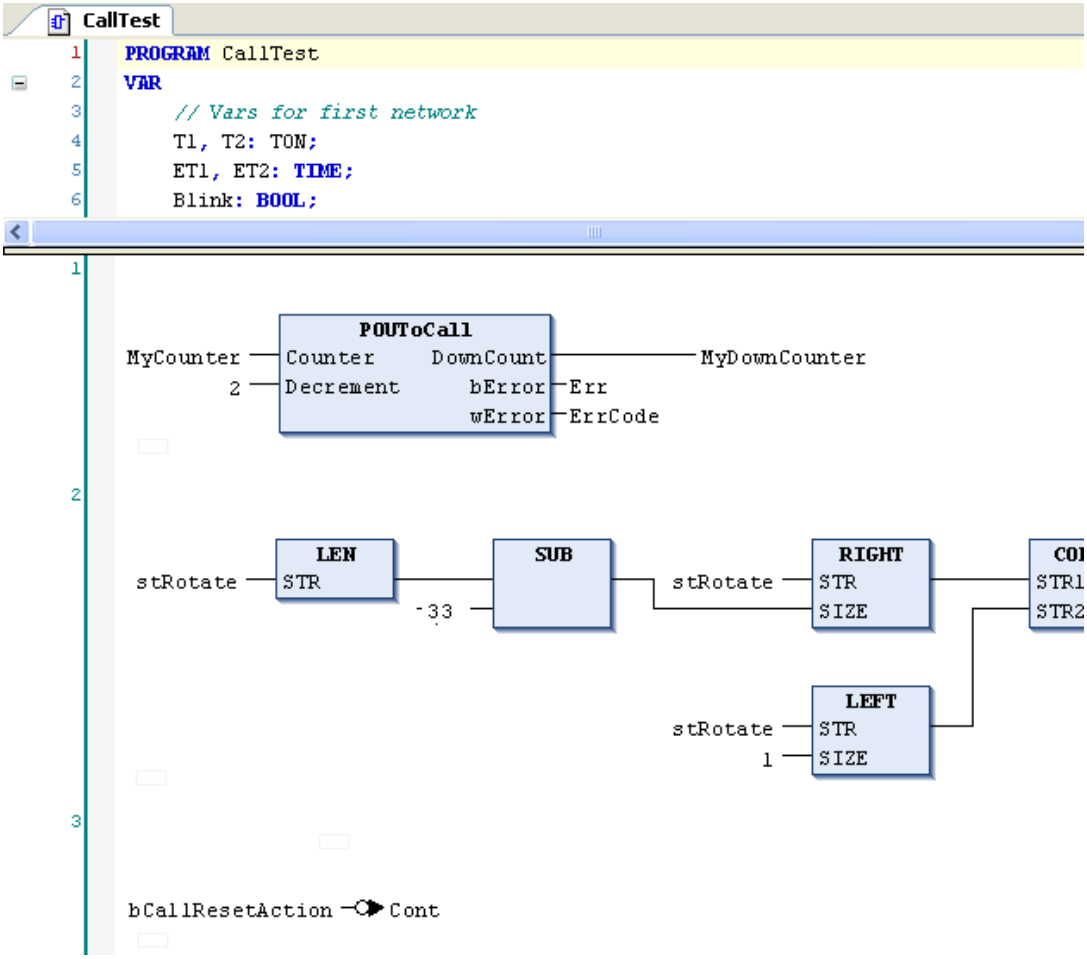
Navigieren im Editor:

- Mit Hilfe der **Pfeiltasten** kann zwischen benachbarten Cursor-Position gesprungen werden, auch netzwerkübergreifend.
- Mit Hilfe der **<Tab>-**Taste kann zur nächsten Cursor-Position innerhalb des Netzwerks gesprungen werden.
- **<Ctrl>+<Home>** setzt den Fokus auf den Dokumentanfang und markiert das erste Netzwerk.
- **<Ctrl>+<End>** setzt den Fokus auf das Dokumentende und markiert das letzte Netzwerk.
- **<PageUp>** blättert eine Seite hoch und markiert das oberste Rechteck.
- **<PageDown>** blättert eine Seite runter und markiert das oberste Rechteck.

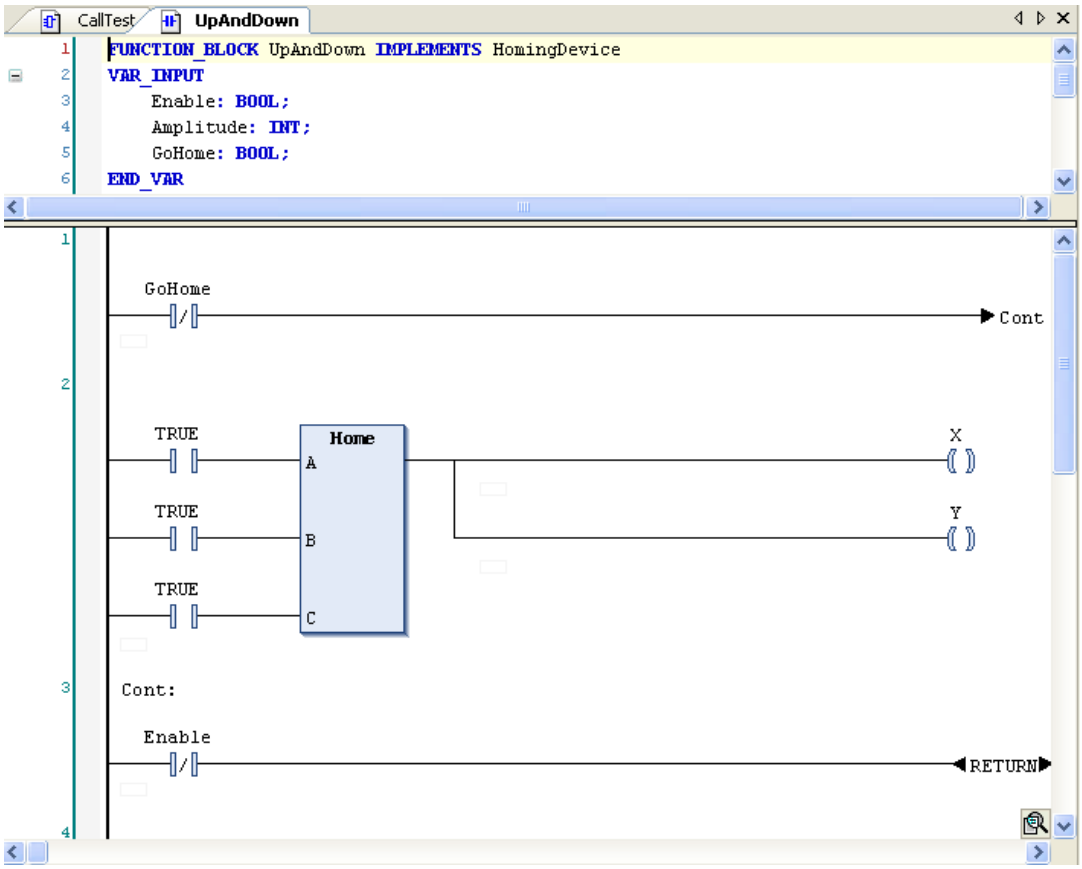
Selektieren (Auswählen):

- Ein Element kann selektiert werden, indem durch Mausklick oder Verwenden der Pfeil- oder der Tabulatortaste die entsprechende Cursor-Position besetzt wird.
- Mehrfachselektion von nicht benachbarten Elementen ist möglich, wenn die <Strg>-Taste gedrückt gehalten wird während die gewünschten Elemente nacheinander selektiert werden.
- Im KOP-Editor wird Mehrfachselektion von benachbarten Elementen erreicht, wenn die <Umschalt>-Taste gedrückt bleibt, während zwei Kontakte ausgewählt werden, die den Beginn und das Ende des gewünschten Netzwerkausschnitts bilden. Wenn ein Netzwerkausschnitt ausgeschnitten (kopiert) und wieder eingefügt werden soll, genügt es, bei gedrückter <Strg>-Taste zwei Kontakte zu selektieren, die den Umfang des Ausschnitts definieren. Dann werden die dazwischen liegenden Elemente automatisch mit berücksichtigt.

Beispiel eines FUP-Editorfensters:



Beispiel eines KOP-Editorfensters:



Informationen zur Programmiersprache siehe:

Funktionsplan - FUP

Kontaktplan - KOP

13.8.4 Arbeiten im AWL-Editor

Der AWL (Anweisungsliste) Editor ist ein **Tabelleneditor** im Unterschied zum reinen Texteditor, der in »PLC Designer« V2.3 verwendet wird. Die **Netzwerk**-Struktur von FUP- oder KOP-Programmen findet sich auch im AWL-Programm wieder. Dabei ist in AWL grundsätzlich ein einziges Netzwerk ausreichend, jedoch ist es im Hinblick auf das Hin- und Herschalten zwischen FUP-, KOP- und AWL-Ansicht eines Programms ggfs. von Vorteil, auch ein AWL-Programm mit Hilfe von Netzwerken bewußt zu strukturieren.

Beachten Sie die generellen Editor-Einstellungen im Optionen-Dialog, Kategorie 'FUP, KOP und AWL'.

Tooltip mit Informationen zu Variablen oder Bausteinparametern: Siehe Modifikatoren und Operatoren in AWL / Tooltip.

Einfügen und Anordnen von Elementen:

- Die **Befehle zum Arbeiten im Editor** sind standardmäßig im FBD/LD/IL Menü enthalten, die wichtigsten stehen auch im Kontextmenü bereit.
- Programmiereinheiten, d.h. **Elemente**, werden jeweils an der aktuellen Cursor-Position mit Hilfe der 'Einfügen'-Befehle eingefügt, die standardmäßig im FUP/KOP/AWL-Menü zu finden sind.
- Die Befehle '**Ausschneiden**', '**Kopieren**', '**Einfügen**' und '**Löschen**', die standardmäßig im Bearbeiten-Menü verfügbar sind, können verwendet werden, um die Elemente anzuordnen.
- Siehe hierzu auch: Cursor-Positionen in FUP, KOP und AWL
- Siehe hierzu auch: finden Sie Informationen zur Programmiersprache AWL.

Sehen Sie im folgenden, wie der AWL Tabelleneditor strukturiert ist, wie man darin navigiert und wie komplexe Operanden, Aufrufe und Sprünge verwendet werden.

Struktur des AWL Tabelleneditors:

Jede Programmzeile steht in einer Tabellenzeile, die durch die folgenden Tabellenspalten in einzelne **Felder** unterteilt wird:

Spalte	enthält..	Beschreibung
1	Operator	Dieses Feld enthält den AWL-Operator (LD, ST, CAL, AND, OR etc.) oder einen Funktionsnamen. Wenn ein Funktionsblock aufgerufen wird, müssen hier zusätzlich die entsprechenden Parameter angegeben werden; im Präfix-Feld muss in diesem Fall ":=" oder "=>" eingegeben werden. Siehe hierzu auch:Modifikatoren und Operatoren in AWL.
2	Operand	Dieses Feld enthält genau einen Operanden oder den Namen einer Sprungmarke. Wenn mehr als ein Operand benötigt wird (erweiterbare Operatoren "AND A, B, C" oder Funktionsaufrufe mit mehreren Parametern), müssen diese in den nächsten Zeilen eingegeben werden, in denen das Operator-Feld leer bleibt. In diesem Fall muss im Postfix-Feld jeweils ein Komma zur Abtrennung zwischen den Parametern eingegeben werden (s.u. Abbildung "AWL Tabelleneditor"). Im Fall eines Aufrufs von Funktionblock, Programm oder Aktion müssen die passenden öffnenden und schließenden Klammern hinzugefügt werden.
3	Adresse	Dieses Feld enthält die Adresse des Operanden, wie sie bei dessen Deklaration definiert wurde. Das Feld kann nicht editiert werden und kann über die Option 'Symboladresse anzeigen' in der Ansicht aus- oder eingeblendet werden.
4	Symbolkommentar	Dieses Feld enthält den Kommentar, der ev. für den Operanden bei dessen Deklaration angegeben wurde. Das Feld kann nicht editiert werden und kann über die Option 'Symbolkommentar anzeigent' in der Ansicht aus- oder eingeblendet werden.
5	Operandenkommentar	Dieses Feld enthält den Kommentar für die aktuelle Programmzeile. Dieser ist hier editierbar. Die Anzeige des Kommentars kann über die Option 'Operandenkommentar'.ein- oder ausgeblendet werden.

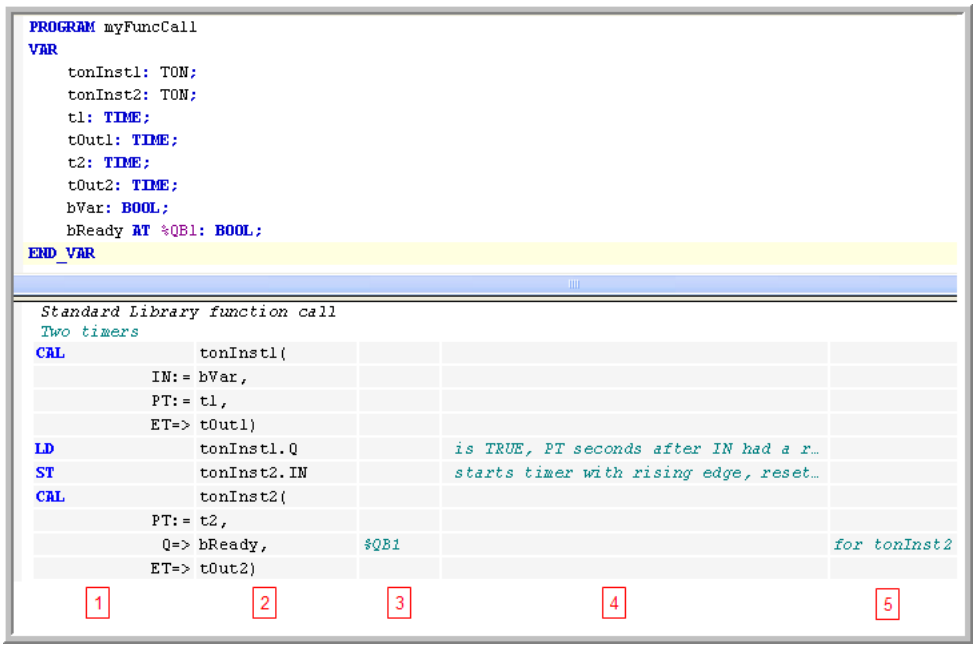


Abb. 368 AWL Tabelleneditor

Navigieren in der Tabelle:

- **Auf**- und **Ab** Pfeiltasten: Springen zum oberhalb bzw. unterhalb liegenden Feld.
- **<Tab>**: Springen zum nächsten Feld innerhalb der Zeile (nach rechts).
- **<Umschalt>** + **<Tab>**: Springen zum vorhergehenden Feld innerhalb der Zeile (nach links).
- **<Leerzeichen>**: Öffnen des Editierrahmens für das gerade ausgewählten Feld. Alternativ kann ein Mausklick auf das Feld ausgeführt werden. Gegebenenfalls steht über die Schaltfläche … der Eingabehilfe-Dialog bereit. Ein geöffnetes Eingabefeld kann mit <Eingabe> geschlossen werden, was die aktuelle Eingabe bestätigt, oder mit <Esc>, was die neuen Eingaben verwirft.
- **<Strg>** + **<Eingabe>**: Einfügen einer neuen Zeile unterhalb der aktuellen.
- **<Entf>**: Löschen der Zeile, in der gerade der Cursor steht.
- **Ausschneiden, Kopieren, Einfügen**: Um eine oder mehrere Zeilen zu kopieren, muss wenigstens ein Feld der Zeile(n) ausgewählt und der 'Kopieren'-Befehl verwendet werden. Um eine Zeile auszuschneiden, wird entsprechend der 'Ausschneiden'-Befehl verwendet. 'Einfügen' fügt die vorher kopierte(n) bzw. ausgeschnittene(n) Zeile(n) oberhalb der aktuellen Zeile ein. Wenn der Cursor aktuell in keiner Zeile steht, wird am Ende des Netzwerks eingefügt.
- **<Ctrl>** + **<Home>** setzt den Fokus auf den Dokumentanfang und markiert das erste Netzwerk.
- **<Ctrl>** + **<End>** setzt den Fokus auf das Dokumentende und markiert das letzte Netzwerk.
- **<PageUp>** blättert eine Seite hoch und markiert das oberste Rechteck.
- **<PageDown>** blättert eine Seite runter und markiert das oberste Rechteck.

Mehrfache Operanden (erweiterbare Operatoren):

Wenn der gleiche Operator mit mehreren Operanden verwendet wird, gibt es zwei mögliche Wege der Programmierung:

1. Die Operanden werden in aufeinanderfolgenden Zeilen eingegeben, getrennt durch Kommata im Postfix-Feld; Beispiel::

```
LD      7
ADD     2,
        4,
        7
ST      iVar
```

2. Die Anweisung wird in aufeinanderfolgenden Zeilen wiederholt, Beispiel:

```
LD      7
ADD     2
ADD     4
ADD     7
ST      iVAR
```

Komplexe Operanden:

Wenn ein komplexer Operand verwendet wird, muss eine öffnende Klammer im Präfix-Feld eingegeben werden, die weiteren Operandeneinträge in den darauf folgenden Zeilen, und die schließende Klammer in einer separaten Zeile, ebenfalls im Präfix-Feld.

Beispiel: Ein String wird rotiert, jeden Zyklus um 1 Zeichen

(entsprechender Code in Strukturiertem Text: stRotate := CONCAT(RIGHT(stRotate, (LEN(stRotate) - 1)), (LEFT(stRotate, 1)));)

```
LD      stRotate
RIGHT(  stRotate
LEN
SUB     1
)
CONCAT( stRotate
LEFT    1
)
ST      stRotate
```

Funktionsaufrufe:

Der Funktionsname wird im Operator-Feld eingegeben. Der (erste) Eingabeparameter muss in einer vorangehenden LD-Operation als Operand gegeben werden. Wenn es weitere Eingabeparameter gibt, muss der nächste in der gleichen Zeile wie der Funktionsname stehen. Weitere können entweder ebenfalls in dieser Zeile, durch Kommata getrennt, oder in weiteren Zeilen hinzugefügt werden.

Der Rückgabewert der Funktion wird in den Akkumulator geschrieben, aber ...

Beachten Sie die folgende Einschränkung hinsichtlich des IEC-Standards: Ein **Funktionsaufruf mit mehreren Rückgabewerten ist nicht möglich**, nur ein einziger Rückgabewert kann für die nachfolgende Operation verwendet werden.

Beispiel: Funktion GeomAverage, die drei Eingabeparameter hat, wird aufgerufen. Der erste Parameter wird als X7 aus einer vorangehenden Operation geliefert, der zweite, "25", steht hinter dem Funktionsnamen. Der dritte wird mit Variable tvar entweder in der selben Zeile oder in der nachfolgenden angegeben.

Der Rückgabewert wird der Variablen 'Ave' zugewiesen:

(entsprechender Code in Strukturiertem Text: Ave := GeomAverage(X7, 25, tvar);)

```
LD          X7
GeomAverage 25
ST          Ave
```

Funktionsblock-Aufrufe, Programmaufrufe:

Verwenden Sie hier den CAL- oder CALC Operator. Tragen Sie den Namen des Funktionsblocksinstanz bzw. der Programmname im Operanden-Feld (zweite Spalte) ein. Die öffnende Klammer wird in der gleichen Zeile im Postfix-Feld eingegeben. Die Eingabeparameter werden einzeln in den nachfolgenden Zeilen eingetragen, wie folgt:

Operator-Feld: Parametername

Präfix-Feld: „:=“ für Eingabeparameter; „=>“ für Ausgabeparameter

Operanden-Feld: aktueller Parameter

Postfix-Feld: „,“ falls weitere Parameter folgen; „)“ nach dem letzten Parameter. Im Fall von parameterlosen Aufrufen muss das Postfix-Feld „()“ enthalten.

Beispiel: Aufruf von POUToCAll mit zwei Ein- und zwei Ausgangsparametern

(entsprechender Code in Strukturiertem Text: POUToCall(Counter := iCounter, iDecrement:=2, bError=>bErr, wError=>wResult);)

```
PROGRAM IL_EXAMPLE
VAR
    bErr: BOOL;
    wResult: WORD;
END_VAR
```

```
CAL     POUToCall(
    iCounter:= 1,
    iDecrement:= 1000,
    wError=> wResult)
LD      POUToCall.bError
ST      bErr
```

Es müssen nicht alle Parameter eines Funktionsblocks oder Programms belegt werden.

Hinweis: Als Einschränkung bezüglich des IEC-Standards können komplexe Ausdrücke nicht verwendet werden. Solche müssen dem Funktionsblock oder dem Programm noch vor dem Aufruf zugewiesen werden.

Aktionsaufruf:

Durchzuführen wie ein Funktionsblock- oder Programmaufruf. Der Aktionsname muss an den Instanznamen oder Programmnamen angehängt werden.

Beispiel: Aufruf von Aktion ResetAction (entsprechender Code in Strukturiertem Text: Inst.ResetAction();)

```
CAL     Inst.ResetAction()
```

Methodenaufruf:

Durchzuführen wie ein Funktionsaufruf. Der Instanzname mit angehängtem Methodennamen muss im Operator-Feld eingegeben werden.

Beispiel: Aufruf von Methode Home (entsprechender Code in Strukturiertem Text: Z := IHome.Home(TRUE, TRUE, TRUE);)

```
LD          TRUE
IHome.Home  TRUE ,
            TRUE
ST          Z
```

Sprung:

Ein Sprung wird durch Verwendung von "JMP" im Operator-Feld und einem Sprungmarken-Namen im Operanden-Feld programmiert. Die Sprungmarke muss im Ziel-Netzwerk im Marken-Feld definiert sein. Beachten Sie, dass die dem unbedingten Sprung vorausgehende Anweisungsfolge mit einem der folgenden Kommandos enden muss: ST, STN, S, R, CAL, RET oder ein weiterer JMP. Das trifft nicht zu für einen bedingten Sprung, der durch den Befehl "JMPC" im Operator-Feld anstelle von "JMP" programmiert wird. Die Ausführung des Sprungs hängt vom geladenen Wert ab.

Beispiel: Bedingte Sprunganweisung; wenn bCallRestAction "TRUE" ist, springt die Programmausführung zum Netzwerk mit der Marke "Cont".

```
LDN     bCallResetAction
JMPC    Cont
```

13.8.5 Cursor-Positionen in FUP, KOP und AWL

AWL-Editor:

Dies ist ein Texteditor, als Tabelle strukturiert. Jede Tabellenzelle ist eine mögliche Cursor-Position. Siehe auch : Arbeiten im AWL-Editor

FUP- und KOP-Editor:

Dies sind grafische Editoren. Siehe unten, Beispiele (1) to (10) zeigen die möglichen Cursor-Positionen: Text, Eingang, Ausgang, Baustein, Kontakt, Spule, Verbindungslinie zwischen den Elementen, Sub-Netzwerk.

Das Setzen des Cursors entspricht dem "Selektieren" eines Elements bzw. Texts. Aktionen wie Ausschneiden, Kopieren, Einfügen, Löschen und andere editorspezifische Befehle können an der aktuellen Cursor-Position bzw. auf das gerade selektierte Element angewendet werden. Siehe auch: Arbeiten im FUP- und KOP-Editor

In FUP wird die aktuelle Cursor-Position durch einen gestrichelten Rahmen um das jeweilige Element angezeigt, Texte und Bausteine, auch Spulen und Kontakte werden zusätzlich blau bzw. rot hinterlegt.

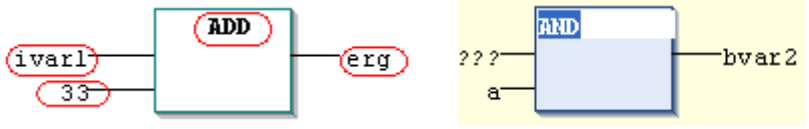
In KOP erscheinen Spulen- und Kontaktsymbole rot, wenn der Cursor darauf platziert ist.

Die Cursor-Position bestimmt, welche Element im Kontextmenü für das Einfügen angeboten werden.

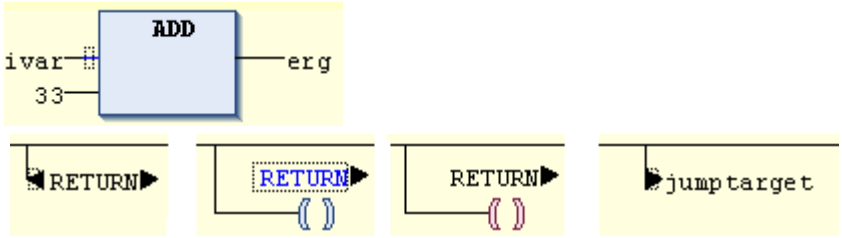
Mögliche Cursor-Positionen:

(1) Jedes Textfeld:

In der folgenden Abbildung sind links die möglichen Cursor-Positionen durch die rote Umrandung angezeigt, rechts wird ein Baustein gezeigt, in dem der Cursor gerade im "AND"-Feld platziert ist. Beachten Sie, dass anstelle der Variablennamen auch die Adressen angezeigt werden können, wenn die entsprechende Option im 'FBD and LD Editor Optionen-Dialog.aktiviert ist.



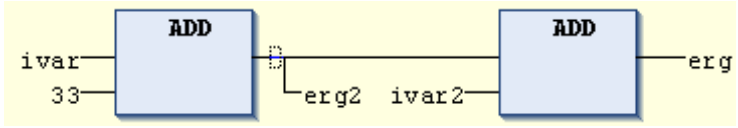
(2) Jeder Eingang:



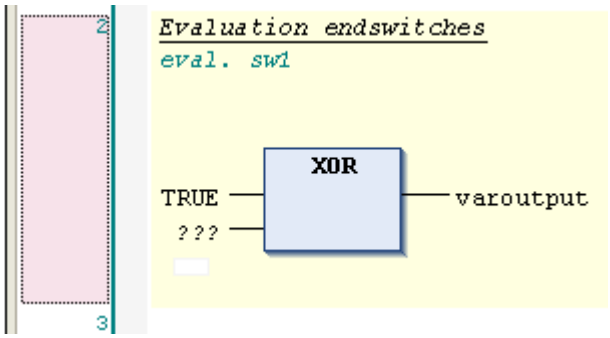
(3) Jeder Operator-, Funktions- oder Funktionsblock-Baustein:

(4) Jeder Ausgang, dem eine Zuweisung oder ein Sprung folgt:

(5) Die Stelle vor einer Verzweigung zu einer Zuweisung, einem Sprung oder einer Return-Anweisung:

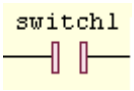


(6) Die am weitesten rechts gelegene Cursor-Position oder Irgendwo im Netzwerk, an keiner der sonstigen Cursor-Positionen . Damit wird das gesamte Netzwerk selektiert.:

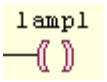


(7) Das Linienkreuz direkt vor einer Zuweisung:

(8) Jeder Kontakt

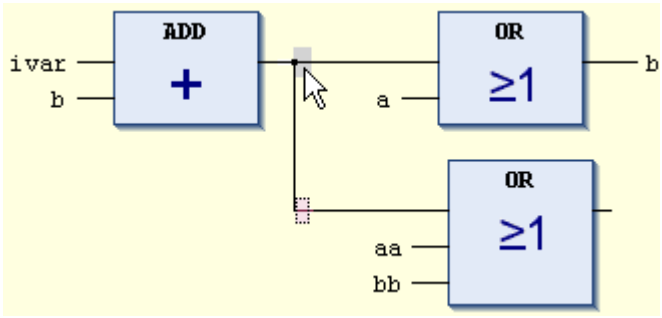


(9) Jede Spule



(10) Die Verbindungslinie zwischen Kontakten und Spulen.

(11) Leitungszweige bzw. "Sub-Netzwerke" innerhalb eines Netzwerks



13.8.6 FUP/KOP/AWL Menü

Wenn der Cursor im FUP/KOP/AWL-Editor steht, ist standardmäßig das FUP/KOP/AWL-Menü in der Menüleiste verfügbar, das die folgenden Befehle zum Programmieren im Editor bereitstellt:

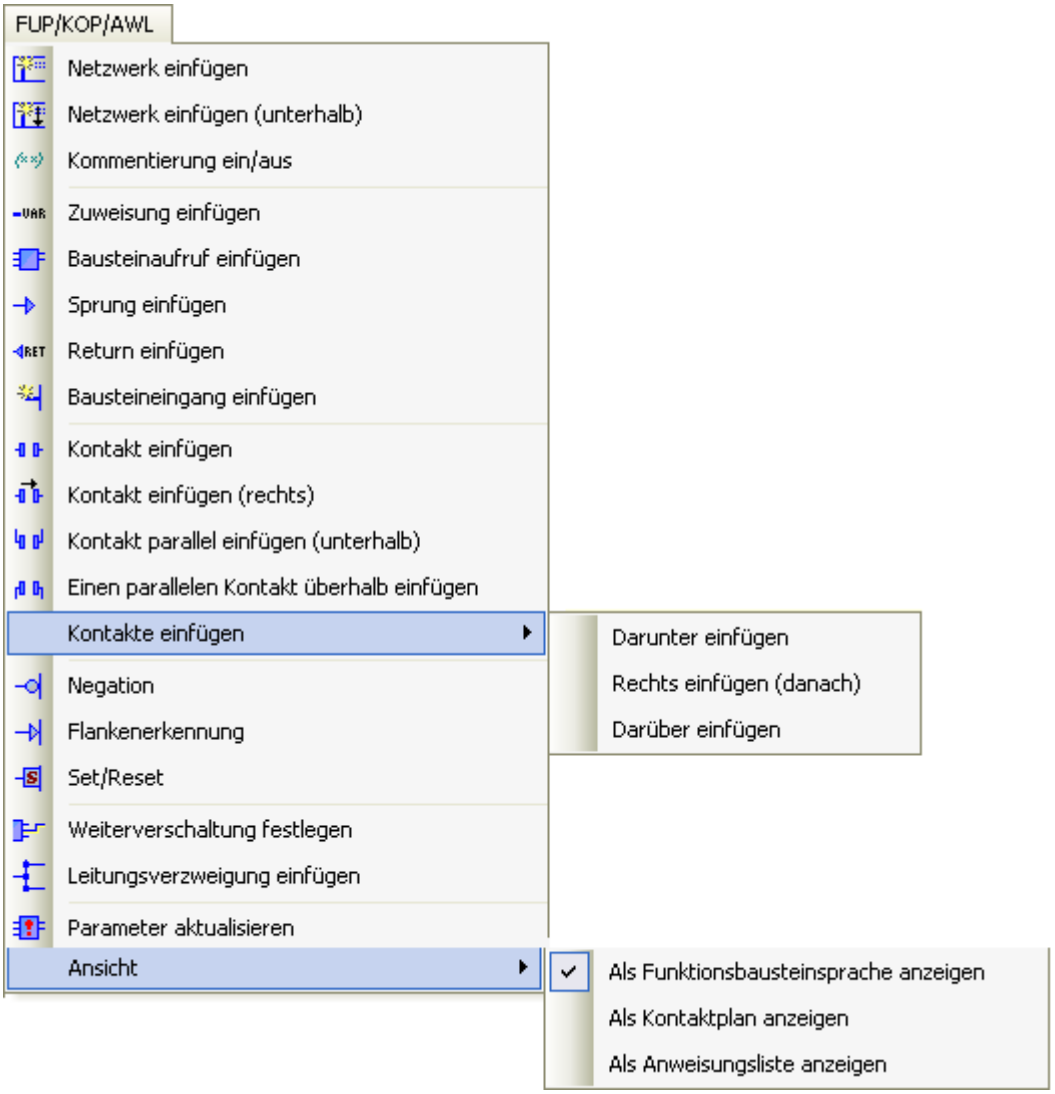


Abb. 369 FUP/KOP/AWL Menü (Standard)

Zur Beschreibung der einzelnen Befehle siehe: FUP/KOP/AWL Editor Befehle.

Wie das Menü gegebenenfalls umkonfiguriert werden kann, siehe unter Anpassen-Dialoge.

13.8.7 FUP/KOP/AWL Elemente

FUP/KOP/AWL Werkzeuge

Der FUP/KOP/AWL-Editor stellt in einer separaten Ansicht eine Werkzeug-Box (Toolbox) bereit, aus der die einzelnen Programmierelemente per "drag&drop" ins Editorfenster eingefügt werden können. Standardmäßig wird die Werkzeug-Box automatisch neben dem Editorfenster geöffnet, kann bei Bedarf auch explizit über den Befehl 'Werkzeuge' im Ansicht-Menü geöffnet werden.

Je nachdem **welche Editoransicht gerade geöffnet ist** können die entsprechenden Elemente in der Werkzeug-Box ausgewählt werden. Nicht verfügbare Elemente sind grau schattiert (Siehe hierzu die Beschreibung der jeweils entsprechenden Einfügen-Befehle).

Die "Werkzeuge" (Elemente) sind in Kategorien sortiert: Generelles (allgemeine Elemente wie Netzwerk, Zuweisung etc.), Logische Operatoren, Mathematische Operatoren, Andere Operatoren (z.B. SEL, MUX, LIMIT and MOVE), Funktionsbausteine (z.B. R_TRIG, F_TRIG, RS, SR, TON, TOF, CTD, CTU), Kontaktplan Elemente und POUs.

Die "POUs" Kategorie bietet alle POUs an, die vom Anwender unter der selben Applikation angelegt wurden wie der gerade im Editor geöffnete FBD/LD/IL-Baustein. Wenn einer POU in seinen Eigenschaften ein Bitmap zugewiesen ist, erscheint dieses vor dem POU-Namen, ansonsten das Standard-Icon zur Kennzeichnung des POU-Typs. Die Liste wird automatisch aktualisiert, wenn POUs unter der Applikation hinzugefügt oder entfernt werden.

Durch Mausklick auf die Schaltfläche, die den Kategorienamen trägt, wird die Kategorie "aufgeklappt", d.h. die Elemente angezeigt. Sehen Sie in der folgenden Abbildung: Kategorie "Generelles" ist gerade aufgeklappt, die anderen sind zugeklappt.

Die Abbildung zeigt ein Beispiel für das Einfügen einer Zuweisung via "Drag&Drop" aus der Werkzeug-Box:

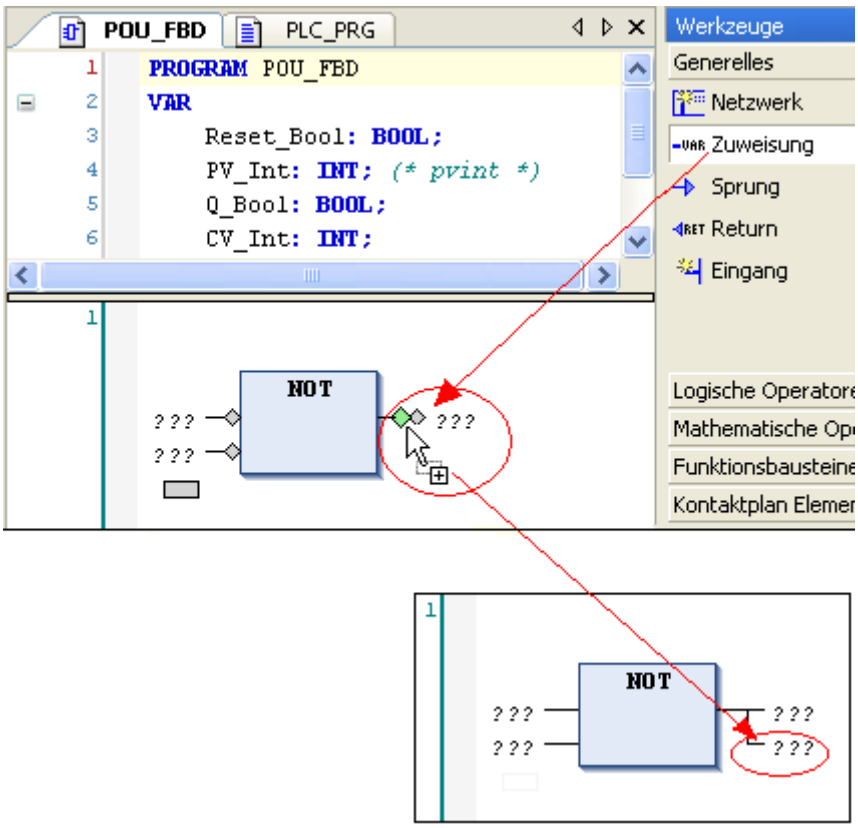


Abb. 370 Einfügen aus der Werkzeug-Box, Beispiel

Um ein Element im Editor **einzufügen**, wählt man es mit einem Mausklick in der Werkzeug-Box aus und zieht es bei weiterhin gedrückter Maustaste ins Editorfenster. Die möglichen Einfügepositionen werden durch **Positionsmarken** (üblicherweise Rautensymbole) angezeigt, solange man das Element mit gedrückter Maustaste über das Editorfenster bewegt. Die nächstgelegene mögliche Einfügeposition leuchtet dabei **grün** auf. Sobald die Maustaste losgelassen wird, wird das Element an der gerade grün angezeigten Position eingefügt.

Wenn ein Baustein-Element auf ein bestehendes gezogen und eingefügt wird, wird das neue das alte ersetzen; wenn dem alten Element bereits Eingänge und Ausgänge zugewiesen waren, werden diese grundsätzlich beibehalten.

Netzwerk in FUP/KOP/AWL

Ein Netzwerk ist die Basiseinheit eines FUP- oder KOP-Programms. Im FUP/KOP/AWL Editor sind die Netzwerke in einer Liste untereinander angeordnet. Jedes Netzwerk ist an der linken Seite mit einer fortlaufenden Netzwerknummer versehen und enthält logische oder arithmetische Ausdrücke, Programm-, Funktions- oder Funktionsblockaufrufe und eine Sprung- oder Return-Anweisung.

Der AWL-Editor, bedingt durch die gemeinsame Editorumgebung mit dem FUP- und KOP-Editor, verwendet ebenfalls das Netzwerk-Element. Wenn eine POU ursprünglich in FUP oder KOP programmiert ist und dann nach AWL konvertiert wird, wird die Netzwerkstruktur beibehalten. Wenn ein Programm ursprünglich in AWL erstellt wird, besteht es aus mindestens 1 Netzwerk, das alle Anweisungen enthalten kann; Es kann jedoch sinnvoll sein, auch das AWL-Programm bereits mit Hilfe von mehreren Netzwerken zu strukturieren, ev. im Hinblick auf eine geplante Konvertierung nach FUP oder KOP.

Ein Netzwerk kann optional mit einem **Titel**, einem **Kommentar** oder einer **Marke** versehen werden.

Die Verfügbarkeit von Titel- und Kommentar-Feldern kann im Dialog 'FBD, LD, IL Editor Options ein- oder ausgeschaltet werden. Ist die Funktion eingeschaltet, so kann das Eingabefeld für den Titel geöffnet werden durch einen Mausklick ins Netzwerk direkt unterhalb der oberen Grenze. Klicken Sie direkt unterhalb des Titels um ein Textfeld zur Eingabe eines Kommentars zu öffnen. Der Kommentar kann mehrzeilig sein. Zeilenumbrüche können mit <Eingabe> eingefügt werden, die Kommentartexteingabe wird mit <Strg>+<Eingabe> abgeschlossen. Ob und wie der Netzwerkkommentar im Editor angezeigt wird, ist im 'FUP-, KOP- und AWL Optionen-Dialog definiert.

Zum Einfügen einer Marke, die als Sprungziel für einen Sprung aus einem anderen Netzwerk verwendet werden kann, dient das Kommando "Sprungmarke einfügen". Nach Definition einer Sprungmarke wird diese unterhalb des Titel- bzw. Kommentar-Feldes angezeigt oder, sollten diese Felder nicht vorhanden sein, direkt unterhalb der oberen Grenze des Netzwerkes.

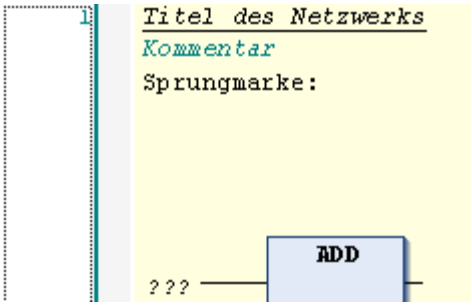


Abb. 371 Position von Titel, Kommentar und Sprungmarke in einem Netzwerk

Ein Netzwerk kann "**auskommentiert**" werden, d.h. das gesamte Netzwerk zum Kommentar umdefiniert (Befehl 'Kommentierung ein/aus') und damit von der Abarbeitung ausgeschlossen werden.

Wenn ein Netzwerk selektiert ist (Cursor-Position 6) können die Standardbefehle zum **Kopieren**, **Ausschneiden**, **Einfügen** und **Löschen** angewendet werden.

Hinweis: Wird ein Klick mit der rechten Maustaste (Cursor-Position 6) auf einem Titel, einem Kommentar oder eine Marke ausgeführt, so wird damit nur dieser Eintrag und nicht das gesamte Netzwerk selektiert. Somit hat das Anwenden der Standardbefehle in diesem Fall keinen Einfluss auf das Netzwerk selbst.

Um ein Netzwerk einzufügen, verwenden Sie den Befehl Netzwerk einfügen oder ziehen Sie das Element aus der Werkzeugbox. Ein Netzwerk mit all seinen Elementen kann auch via "Drag&Drop" innerhalb des Editors kopiert oder verschoben werden.

Beachten Sie die Möglichkeit, durch das Einfügen von Verzweigungen "Subnetzwerke" zu erzeugen.

RET Netzwerk:

Im Online-Modus wird automatisch ein zusätzliches, leeres Netzwerk unterhalb der vorhandenen Netzwerke dargestellt. Anstelle einer Netzwerknummer ist es mit "RET" bezeichnet. Es repräsentiert die Position, an der bei der Abarbeitung zum aufrufenden Baustein zurückgekehrt wird und bietet eine mögliche Haltepunktposition.

Zuweisung in FUP/KOP/AWL

In FUP oder KOP wird eine Zuweisung abhängig von der aktuellen Cursor-Position entweder direkt vor einem Eingang (Cursor-Position 2), direkt nach einem Ausgang (Cursor-Position 4) oder am Ende des Netzwerks (Cursor Position 6) als Linie eingefügt. In einem KOP-Netzwerk wird eine Zuweisung als Spule eingefügt. Alternativ kann das Zuweisungselement aus der Werkzeugbox gezogen werden oder via Drag&Drop innerhalb des Editorfensters kopiert oder verschoben werden.

Nach dem Einfügen kann die Zeichenfolge "???" durch den Namen der Variablen, die zugewiesen werden soll, ersetzt werden. Die Eingabehilfe kann dabei über Schaltfläche zu Hilfe genommen werden.

In AWL wird eine Zuweisung durch Verwenden der "KOP"- und "ST"- Anweisungen programmiert. Siehe hierzu auch: Modifikatoren und Operatoren in AWL.

Sprung in FUP/KOP/AWL

In FUP oder KOP wird ein Sprung abhängig von der aktuellen Cursor-Position entweder direkt vor einem Eingang (Cursor-Position 2), direkt nach einem Ausgang (Cursor-Position 4) oder am Ende des Netzwerks (Cursor Position 6) als Linie eingefügt. Alternativ kann das Sprungelement aus der Werkzeugbox gezogen werden oder via Drag&Drop innerhalb des Editorfensters kopiert oder verschoben werden.

Nach dem Einfügen kann die Zeichenfolge "???" durch den Namen der Marke ersetzt werden, die als Sprungziel dienen soll, d.h. die das Netzwerk, zu dem gesprungen werden soll, bezeichnet.

In AWL wird ein Sprung durch Verwenden einer "JMP"-Anweisung programmiert. Siehe in diesem Zusammenhang auch: Modifikatoren und Operatoren in AWL

Marke in FUP/KOP/AWL

Jedes FUP oder KOP-Netzwerk hat unterhalb des Feldes für den Netzwerkkommentar ein Texteingabefeld, in dem eine Marke definiert werden kann. Diese Marke ist eine optionaler Kennzeichner für das Netzwerk und kann als Adresse für einen Sprung angegeben werden. Sie kann aus einer beliebigen Zeichenfolge bestehen.

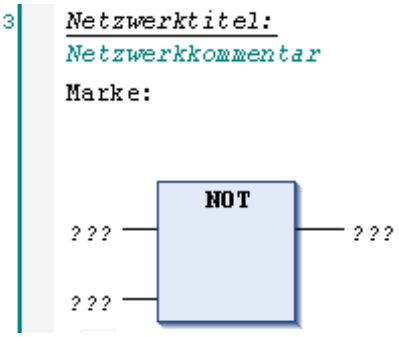


Abb. 372 Marke für Netzwerk

Verwenden Sie den 'FUP-, KOP- und AWL Optionen-Dialog, um die Anzeige von Netzwerktitel und -kommentar zu definieren.

Bausteine in FUP/KOP/AWL

Ein Baustein, der bzw. dessen Aufruf in einem FUP-, KOP- oder AWL-Netzwerk eingefügt wird, ist ein komplexes Element und kann zusätzliche Funktionen repräsentieren, wie z.B. Timer, Zähler, arithmetische Operationen oder auch Programme, IEC-Funktionen und IEC-Funktionsbausteine.

Ein solcher Baustein kann beliebige Ein- und Ausgänge haben und entweder aus einer Bibliothek oder direkt aus dem Projekt stammen. Mindestens ein Eingang und ein Ausgang müssen allerdings boolsche Werte liefern.

Wenn der Baustein eines mitliefert und wenn die Option Bausteinsymbol anzeigen aktiviert ist, wird ein Symbol innerhalb des Bausteinelements angezeigt.

Verwendung in FUP und KOP

Ein Bausteinaufruf kann mit dem Befehl Bausteinaufruf einfügen, Leeren Baustein einfügen im linken Teil eines KOP-Netzwerks (wie eine Spule) bzw. in einem FUP-Netzwerk eingefügt werden. Alternativ kann er aus der Werkzeug-Box gezogen werden oder innerhalb des Editors von einer anderen Position (Drag&Drop). Siehe Befehl Baustein einfügen für Details.

Verwendung in AWL:

In einem AWL Programm wird eine CAL-Anweisung mit Parametern eingefügt, um einen Baustein aufzurufen.

Eine **Aktualisierung** der Bausteinparameter (Eingänge, Ausgänge) im aktuellen Code - für den Fall, dass sich die Baustein-Schnittstellen geändert haben - erhält man mit dem Befehl Parameter aktualisieren', ohne dass der Baustein neu eingefügt werden muss.

RETURN Anweisung in FUP/KOP/AWL

Mit einer RETURN Anweisung kann ein FUP, KOP oder AWL Baustein verlassen werden.

In einem FUP oder KOP Netzwerk kann die RETURN Anweisung parallel zu oder anschließend an die vorausgehenden Elemente platziert werden. Sobald der Eingang der RETURN Anweisung TRUE wird, wird die Abarbeitung des Bausteins sofort abgebrochen.

Zum Einfügen siehe Befehl Return einfügen. Alternativ kann das Element aus der Werkzeugbox gezogen werden oder via Drag&Drop innerhalb des Editorfensters kopiert oder verschoben werden.

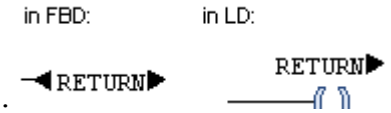


Abb. 373 RETURN Element

In AWL wird zum gleichen Zweck die RET Anweisung verwendet.

Leitungsverzweigung / Nachverschaltung in FUP/KOP/AWL (Implementierung aufgeschoben)

FUP, KOP

In einem FUP- bzw. KOP-Netzwerk spaltet eine Leitungsverzweigung bzw. Nachverschaltung die Abarbeitungslinie ab der aktuellen Cursor-Position. Sie wird in zwei Armen (Subnetzwerken) weitergeführt, die von oben nach unten nacheinander ausgeführt werden. Jedes Subnetzwerk kann weiter verzweigt werden, wodurch innerhalb eines Netzwerks Mehrfachverzweigungen möglich sind.

Jedes Subnetzwerk erhält ein Marker-Symbol (Rechteck), das ausgewählt werden kann (Cursor-Position 11), um Aktionen wie beispielsweise Ausschneiden und Einfügen auf das Subnetzwerk auszuführen.

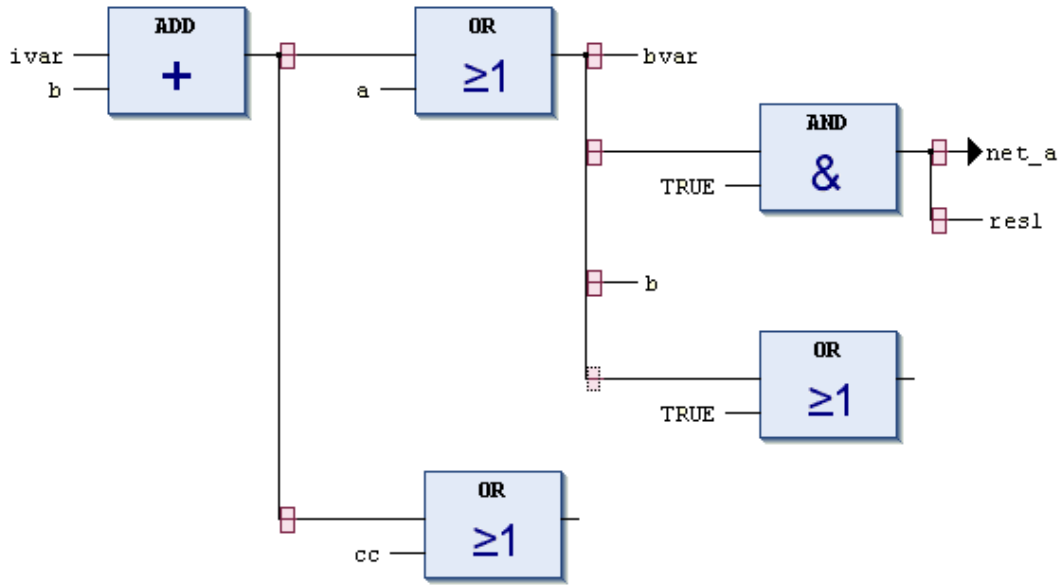


Abb. 374 Beispiel für die Markierungen von Netzwerken innerhalb eines Netzwerks

In FBD wird eine Verzweigung mit Befehl 'Leitungsverzweigung einfügen' eingefügt. Alternativ dazu kann das Element aus der Werkzeugbox gezogen werden. Die möglichen Einfügepositionen sehen Sie unter 'Verzweigung einfügen".

Hinweis: Ausschneiden/Kopieren&Einfügen ist derzeit für Subnetzwerke nicht möglich.

Siehe das in der nachfolgenden Abbildung dargestellte Beispiel: Eine Verzweigung wurde am Ausgang des SUB-Bausteins eingefügt. Dies erzeugt zwei Subnetzwerke, jedes an seinem Subnetzmarker selktierbar. Danach wurde in jedem Subnetzwerk ein ADD-Baustein hinzugefügt.

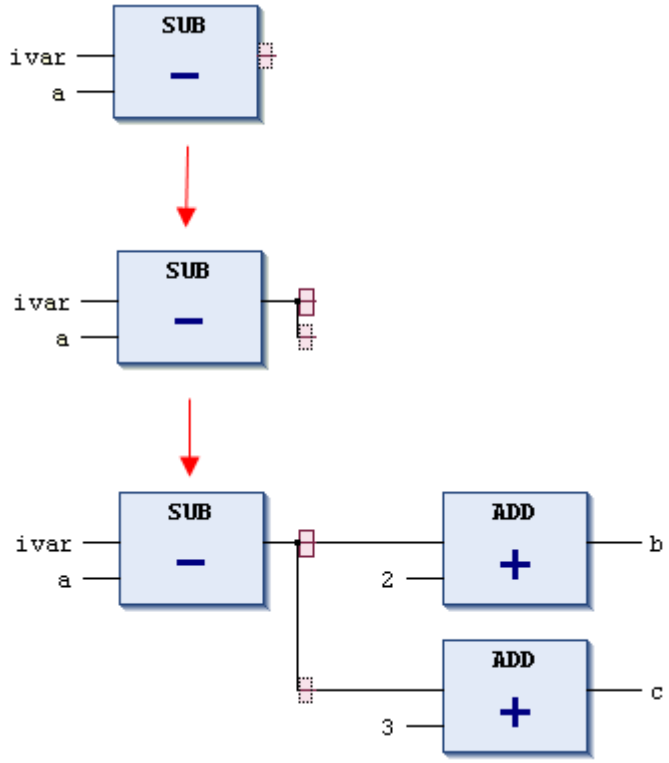


Abb. 375 Beispiel in FUP, Einfügen einer Verzweigung

Um ein Subnetzwerk zu löschen, müssen erst alle seine Elemente gelöscht werden, d.h. alle Elemente rechts vom Subnetzmarker. Dann kann der Marker selektiert werden und der Standardbefehl Löschen bzw. <Entf> verwendet werden.

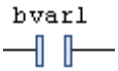
AWL (Anweisungsliste)

In AWL wird eine "Leitungsverzweigung" bzw. "Nachverschaltung" durch eine entsprechende Abfolge von Anweisungen wiedergegeben.

Kontakt

Dies ist ein reines **KOP-Element**.

Jedes Netzwerk in KOP enthält in seinem linken Teil einen oder mehrere Kontakte. Ein Kontakt wird folgendermaßen dargestellt:

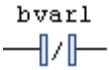


Ein Kontakt gibt die Bedingung "ON" (TRUE) oder "OFF" (FALSE) von links nach rechts weiter, so dass sie schließlich eine Spule im rechten Teil des Netzwerks erreicht. Zu diesem Zweck wird dem Kontakt eine **boolsche Variable** zugewiesen, die die Bedingung enthält.

Mehrere Kontakte können sowohl **in Reihe** als auch **parallel** angeordnet werden. Bei zwei parallelen Kontakten muss nur einer den Wert TRUE erhalten, damit "ON" nach rechts weitergegeben wird. Wenn Kontakte **hintereinander (in Reihe)** geschaltet sind, müssen alle Kontakte den Wert TRUE erhalten, damit vom letzten Kontakt der Reihe "ON" nach rechts weitergegeben wird.

Somit kann mit KOP eine elektrische Parallel- bzw. Reihenschaltung programmiert werden.

Ein Kontakt kann auch **negiert** sein, erkennbar am Schrägstrich im Kontaktsymbol:



Ein negierter Kontakt gibt die eingehende Bedingung (TRUE oder FALSE) nur weiter, wenn die ihm zugewiesene boolsche Variable den Wert FALSE hat. Die Werkzeuge-Box bietet direkt negierte Kontakt-Elemente.

Das **Einfügen** eines Kontakt-Elements in einem KOP-Netzwerk erfolgt entweder mit Hilfe der Menübefehle (Kontakt einfügen, Kontakt einfügen (rechts), Kontakt parallel einfügen, Kontakt parallel einfügen (unterhalb)). Alternativ kann er aus der Werkzeug-Box gezogen werden oder innerhalb des Editors von einer anderen Position (Drag&Drop).

FUP, AWL:

Wenn gerade in der FUP oder AWL Ansicht des Editors gearbeitet wird, stehen die Befehle zum Einfügen eines Kontakts nicht zur Verfügung. Kontakte, die zuvor in der KOP-Ansicht eingefügt wurden, werden jedoch durch die entsprechenden FUP-Elemente bzw. AWL-Anweisungen dargestellt.

Spule

Dies ist ein reines **KOP-Element**.

Auf der rechten Seite eines KOP-Netzwerks können beliebig viele Spulenelemente eingefügt sein. Eine Spule wird folgendermaßen dargestellt:

Mehrere Spulen können nur **parallel** zueinander angeordnet werden. Eine Spule gibt den Wert, der von links geliefert wird, nach rechts weiter und kopiert ihn in die ihr zugewiesene **boolesche Variable**. Ihr Eingang kann den Wert "ON" (TRUE) oder "OFF" (FALSE) haben.

Eine Spule kann **negiert** sein, erkennbar am Schrägstrich im Symbol:

In einer negierten Spule wird der negierte Wert des eingehenden Signals in die der Spule zugewiesenen boolsche Variable kopiert. Somit wird eine negierte Spule nur ein "ON"-Signal weitergeben, wenn diese Variable den Wert FALSE hat.

Eine Spule kann mit Hilfe des Befehls Zuweisung einfügen eingefügt werden, der standardmäßig im FUP/KOP/AWL Menü zu finden ist. Alternativ kann sie aus der Werkzeug-Box ('Kontaktplan Elemente') gezogen werden oder innerhalb des Editors von einer anderen Position (Drag&Drop). Siehe hierzu auch: Set/Reset in FUP/KOP/AWL.

FUP/AWL:

Wenn gerade in der FUP oder AWL Ansicht des Editors gearbeitet wird, stehen die Befehle zum Einfügen eines Kontakts nicht zur Verfügung. Kontakte, die zuvor in der KOP-Ansicht eingefügt wurden, werden jedoch durch die entsprechenden FUP-Elemente bzw. AWL-Anweisungen dargestellt.

Set/Reset in FUP/KOP/AWL

FUP und KOP

Ein boolscher Ausgang in FUP oder entsprechend eine Spule in KOP kann "gesetzt" (set) oder "zurückgesetzt" (reset) werden. Um zwischen dem gesetzten und zurückgesetzten Status zu wechseln verwendet man den Befehl Set/Reset aus dem Kontextmenü, wenn der Ausgang selektiert ist. Der Ausgang bzw. das Spulensymbol werden mit S (set) bzw. R (reset) gekennzeichnet. Siehe Set/Reset zum Einfügen.

Set: Wenn der Wert TRUE an einem gesetzten Ausgang oder einer gesetzten Spule ankommt, werden Ausgang bzw. Spule ebenfalls TRUE und bleiben TRUE. Der Wert kann an dieser Stelle nicht mehr überschrieben werden, solange die Applikation läuft.

Reset: Wenn der Wert TRUE an einem zurückgesetzten Ausgang oder einer zurückgesetzten Spule ankommt, werden Ausgang bzw. Spule FALSE und bleiben FALSE. Der Wert kann an dieser Stelle nicht mehr überschrieben werden, solange die Applikation läuft.

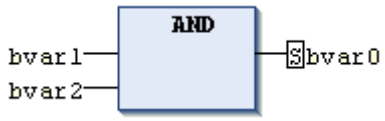


Abb. 376 Gesetzter Ausgang in FUP, Beispiel

Im KOP-Editor können Set- und Reset-Spulen direkt mit drag&drop aus der Werkzeuge-Box, Kategorie 'Kontaktplan-Elemente' eingefügt werden: , .

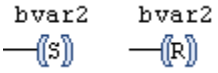


Abb. 377 Gesetzte Spule (Set-Spule), Zurückgesetzte Spule (Reset-Spule), Beispiele

AWL

In einer Anweisungsliste werden die Operatoren S und R verwendet, um einen Operanden zu setzen (Set) oder zurückzusetzen (Reset).

Set/Reset Coil Spulen

Spulen können auch als Set- oder Reset-Spulen definiert werden. Man erkennt eine Set-Spule am eingefügten "S" im Spule-Symbol: **(S).** Eine Set-Spule überschreibt niemals den Wert TRUE der zugehörigen Booleschen Variable, d.h. eine Variable mit Wahrheitswert TRUE behält diesen Wert.

Man erkennt eine Reset-Spule am eingefügten "R" im Spule-Symbol: **(R).** Eine Reset-Spule überschreibt niemals den Wert FALSE der zugehörigen Booleschen Variable, d.h. hat diese Variable den Wahrheitswert FALSE, so bleibt ihr Wert FALSE.

Im KOP-Editor können Set und Reset-Spulen mit Hilfe der Maus direkt aus der Toolbox (Kategorie 'Kontaktplan Elemente': , .)in den Editor gezogen werden.

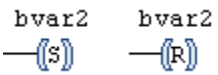


Abb. 378 Beispiel - Set-Spule, Reset-Spule

13.8.8 FUP/KOP/AWL Editor im Online-Modus

Im Online-Modus des FUP/KOP/AWL Editors stehen Ansichten für das Monitoring und für das Schreiben und Forcen von Variablen und Ausdrücken auf der Steuerung bereit. Debugging-Funktionalität (Haltepunkte, schrittweise Abarbeitung etc.) ist ebenfalls enthalten.

- Eine Beschreibung, wie Objekte im Online-Modus geöffnet werden, finden Sie unter Benutzeroberfläche im Online-Modus.
- Informationen zum im oberen Teil des FUP/KOP/AWL-Editors enthaltenen Deklarationseditor im Online-Modus finden Sie Deklarationseditor im Online-
- Modus.

Monitoring

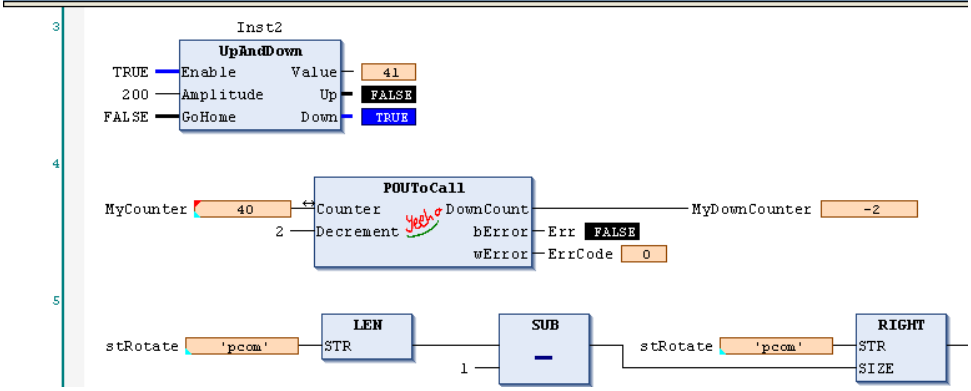
Wenn Inline Monitoring nicht explizit deaktiviert ist (Optionen), wird es in der FUP- und KOP-Ansicht des Editors in Form von kleinen Monitoring-Fenstern hinter jeder Variable bereitgestellt, bzw. in einer separaten Spalte im AWL-Tabelleneditor. Diese Fenster zeigen jeweils den aktuell auf dem Zielsystem vorliegenden Wert (**Inline Monitoring**). Dies gilt auch für nicht zugewiesene Funktionsblockein- und ausgänge.

Das Inline-Monitoring Fenster einer Variable zeigt ein kleines rotes Dreieck in der oberen linken Ecke, wenn die Variable gerade geforced ist, ein blaues in der unteren linken Ecke, wenn die Variable gerade geforced und für das Aufheben des Forces vorbereitet ist. :

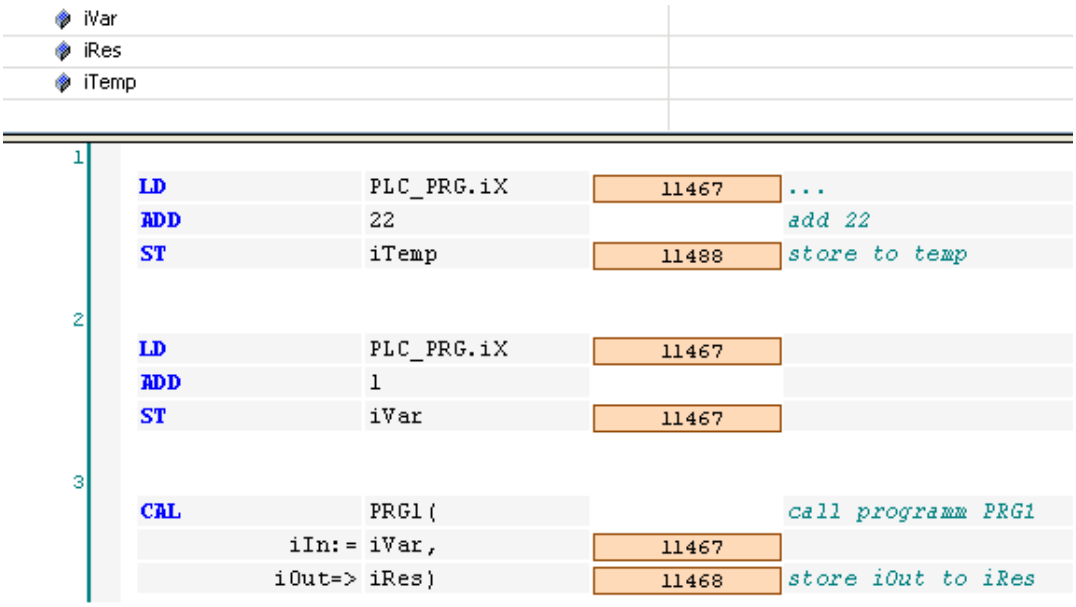
Beispiel: Online-Ansicht eines FUP-Programms:

SP_for_Win32.App.CallTest

Ausdruck	Kommentar	Datentyp	Wert	Vorbereiteter Wert
Inst2		UpAndDown		
Enable		BOOL	TRUE	
Amplitude		INT	200	
GoHome		BOOL	FALSE	
Value		INT	41	
Up		BOOL	FALSE	
Down		BOOL	TRUE	
Counter		DINT	2159	
diValue		DINT	41	
X		BOOL	FALSE	
X		INT	0	
X2		INT	0	
Up		BOOL	TRUE	
Down		BOOL	FALSE	
MyCounter	Vars for third network	DINT	F 40	<Unforce and restore>
MyDownCounter		DINT	-2	
Err		BOOL	FALSE	
ErrCode		WORD	0	
stRotate	Vars for fifth network	STRING	'pcom'	'abc'
Ave	Vars for fifth network	DINT	38	

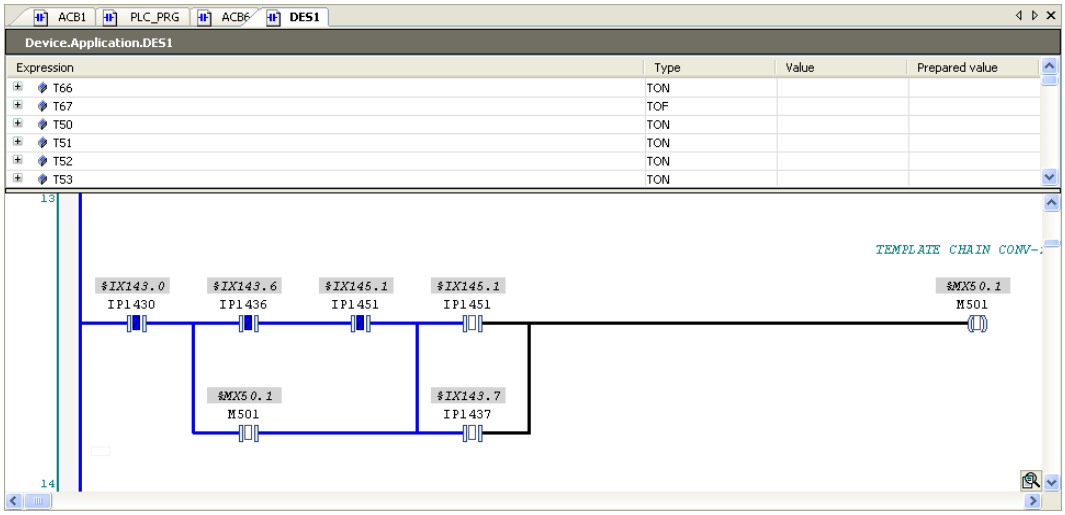


Beispiel: Online-Ansicht eines AWL-Programms:



In der Online-Ansicht eines Kontaktplans (KOP) sind die Anschlussleitungen farblich animiert: Verbindungen mit Wert TRUE werden als fette blaue Linie dargestellt, Verbindungen mit Wert FALSE als fette schwarze Linie, wohingegen Verbindungen von unbekanntem oder analogem Wert normal dargestellt werden (dünne schwarze Linie). (Beachten Sie: der Wert der Verbindungen wird aus den gemonitorten Variablen berechnet, es handelt sich nicht um eine echte Ablaufkontrolle.)

Beispiel - Online-Ansicht eines KOP-Programms:



Beachten Sie für die Online-Ansicht einer Funktionsbaustein-POU: In der Implementierungs-Ansicht zeigen die Monitoring-Fenster im Deklarationsteil keine Variablenwerte an sondern "<Der Wert des Ausdrucks kann nicht gelesen werden.>", und die Inline Monitoring Felder in den Netzwerken zeigen drei Fragezeichen "<???>"

Forcen/Schreiben von Variablen

Im Online-Modus kann eine Variable im Deklarationsteil oder im Implementierungsteil für das Forcen oder Schreiben eines Werts vorbereitet werden. Im Implementierungsteil wird dazu bei einem Mausklick auf die Variable der folgende Dialog geöffnet:

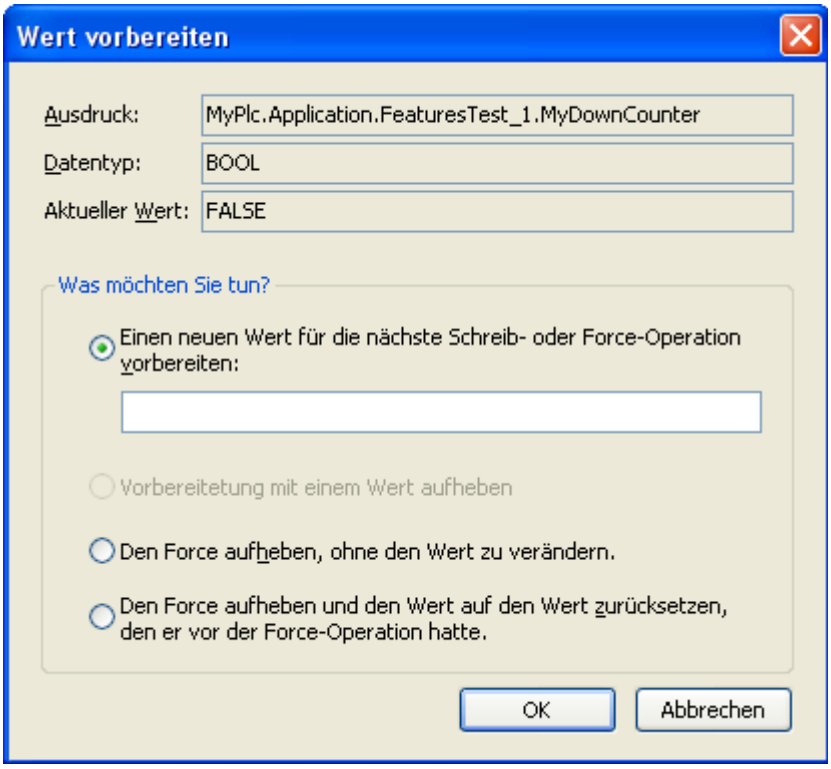


Abb. 379 Dialog - Wert vorbereiten

Der Name, ergänzt mit dem Pfad der Variablen innerhalb des Gerätebaums ("Ausdruck"), wie auch Typ und aktueller Wert der Variablen werden angezeigt. Durch Aktivierung des entsprechenden Punkts können Sie entscheiden, ob

- ein neuer Wert vorbereitet werden soll, der dazu im Editierfeld eingetragen werden muss.
- ein vorbereiteter Wert gelöscht werden soll.
- eine aktuell geforcte Variable freigegeben werden soll.
- eine aktuell geforcte Variable freigegeben und auf ihren ursprünglichen Wert vor dem Forcen gesetzt werden soll.

Die ausgewählte Aktion wird nach Aufruf des Kommandos **Werte forcen** (standardmäßig im Menü Online) oder nach Drücken der Taste **F7** ausgeführt.

Haltepunkt (Breakpoint)- bzw. Haltepositionen:

Mögliche Positionen, die für einen Breakpoint (Haltepunkt) zu Debug-Zwecken gewählt werden können, sind grundsätzlich diejenigen Positionen, an denen sich Variablenwerte ändern können (Anweisungen), an denen sich das Programm verzweigt, oder an denen ein anderer Baustein aufgerufen wird. Also folgende Positionen:

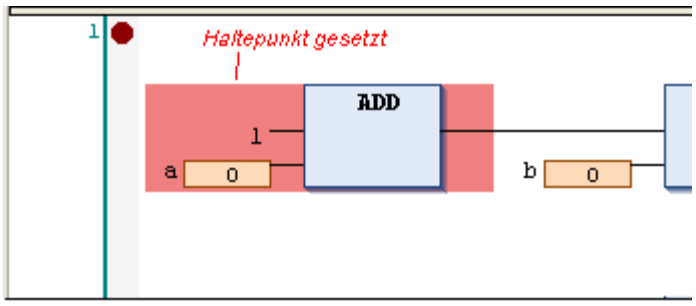
- auf dem gesamten Netzwerk, was bewirkt, dass der Haltepunkt an der ersten möglichen Position im Netzwerk gesetzt wird
- auf einer Bausteinbox, sofern der Baustein eine Zuweisung beinhaltet; also nicht möglich bei Operatoren-Bausteinen wie beispielsweise ADD, DIV); siehe aber nachfolgenden Hinweis
- auf Zuweisungen
- am Ende des Bausteins an der Position der Rückkehr zum aufrufenden Bausteins; im Online-Modus erscheint automatisch ein leeres Netzwerk an dieser Stelle, das anstelle einer Netzwerknummer mit "RET" gekennzeichnet ist.

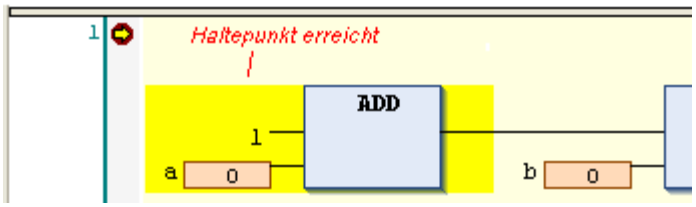
Hinweis: Gegenwärtig kann auf den ersten Baustein im Rechteck nicht direkt ein Haltepunkt gesetzt werden. Wenn jedoch ein Haltepunkt auf das gesamte Netzwerk gesetzt wird, wird diese Haltepositionsmarkierung im Online-Betrieb automatisch auf den ersten Baustein übertragen.

Die aktuell möglichen Positionen können in der Auswahlliste im Haltepunkte-Dialog nachgesehen werden.

Ein Netzwerk, das einen aktiven Haltepunkt enthält, wird mit dem Haltepunkt-Symbol (roter gefüllter Kreis) rechts neben der Netzwerknummer und durch die Hinterlegung der ersten möglichen Haltepunktposition im Netzwerk mit einem rot-schattierten Rechteck markiert. Deaktivierte Haltepunktpositionen werden durch einen nicht gefüllten roten Kreis bzw. eine rote rechteckige Umrahmung angezeigt.

Beispiel: Haltepunkt gesetzt bzw. erreicht:

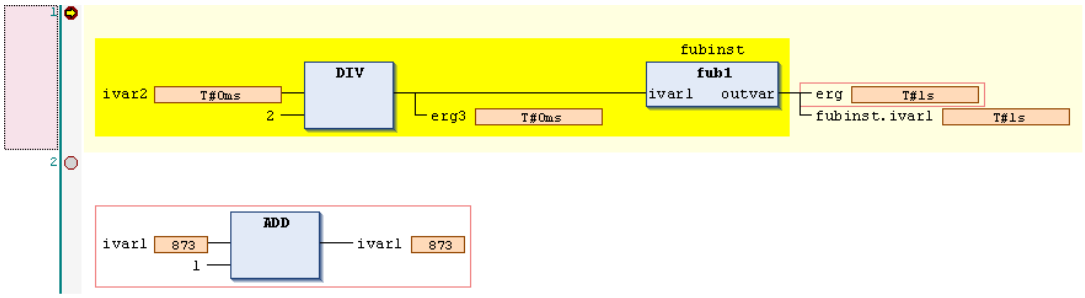




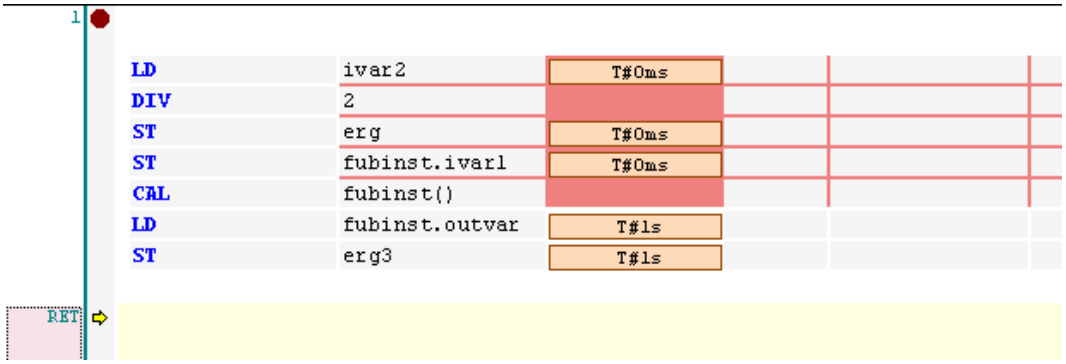
Sobald während der schrittweisen Abarbeitung oder im Programmablauf eine Haltepunkt-Position erreicht wird, erscheint ein gelber Pfeil im Haltepunkt-Symbol und die Rot-Schattierung wechselt nach Gelb.

Haltepositionen, gezeigt in einem FUP-Beispiel:

Die gerade erreichte Halteposition ist gelb hinterlegt, die nachfolgende, noch nicht erreichte rot:



Haltepositionen, gezeigt in einem AWL-Beispiel:



Hinweis: Beachten Sie für Haltepunkte in Methoden: Ein Haltepunkt wird automatisch in allen Methoden, die aufgerufen werden können, gesetzt. Wenn also eine durch eine Schnittstelle verwaltete Methode aufgerufen wird, werden Haltepunkte in allen Methoden gesetzt, die in Funktionsbausteinen (Funktionsblöcken) vorkommen, die diese Schnittstelle implementieren; ebenso in allen abgeleiteten Funktionsbausteinen, die für die Methode "zeichnen". Wenn eine Methode durch einen Zeiger auf einen Funktionsbaustein aufgerufen wird, werden die Haltepunkte in der Methode des Funktionsbausteins und in allen abgeleiteten Funktionsbausteinen, die für die Methode zeichnen.

13.9 Globale-Variablenlisten-Editor

Der Editor für Globale Variablenlisten (GVL-Editor) ist ein Deklarationseditor zum Erstellen und Bearbeiten von Globalen Variablenlisten. Er arbeitet gemäß der aktuellen Einstellungen in den Texteditor-Optionen und zeigt sich im Online-Modus wie es für den Deklarationseditor beschrieben ist. Die Deklaration muss mit "VAR_GLOBAL" beginnen und mit "END_VAR" enden. Diese Schlüsselwörter sind bereits automatisch vorhanden. Dazwischen müssen gültige Deklarationen von globalen Variablen eingefügt werden.

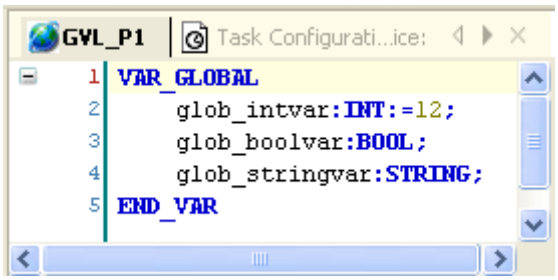


Abb. 380 GVL-Editor

13.10 Bibliotheksverwalter

Der Bibliotheksverwalter dient dem Einbinden und Verwalten von Bibliotheken im Projekt. Die Installation von Bibliotheken wie auch die Definition von Bibliotheks-Ablageorten (Repositories) wird im Bibliotheks-Repository-Dialog vorgenommen. Dieser kann über den gleichnamigen Befehl im Tools-Menü (Standard) oder im Bibliotheksverwalter-Dialog geöffnet werden.

Der Bibliotheksverwalter wird mit dem Plug-In 'Library Manager Editor' bereitgestellt und kann im Projekt über den Objekt hinzufügen Dialog eingefügt werden, entweder global zugänglich im POUs-Fenster oder geräte- bzw. applikationsspezifisch im Geräte-Fenster. An jeder dieser Positionen kann jeweils nur 1 Bibliotheksverwalter pro Ebene eingefügt werden. Der Eintrag trägt den im 'Objekt hinzufügen'-Dialog vergebenen Namen.

Abb. 381 Bibliotheksverwalter Objekt im Geräte-Fenster

Der Bibliotheksverwalter wird über den Befehl Objekt bearbeiten oder durch einen Doppelklick auf den Objekteintrag in einem Editor-Fenster geöffnet.

Übersetzungsfehler bezüglich des Bibliotheksverwalters werden in einer eigenen Kategorie im Meldungsfenster ausgegeben.

13.10.1 Editorfenster des Bibliotheksverwalters

Der Bibliotheksverwalter wird über den Befehl Objekt bearbeiten oder durch einen Doppelklick auf den Objekteintrag in einem Editor-Fenster geöffnet.

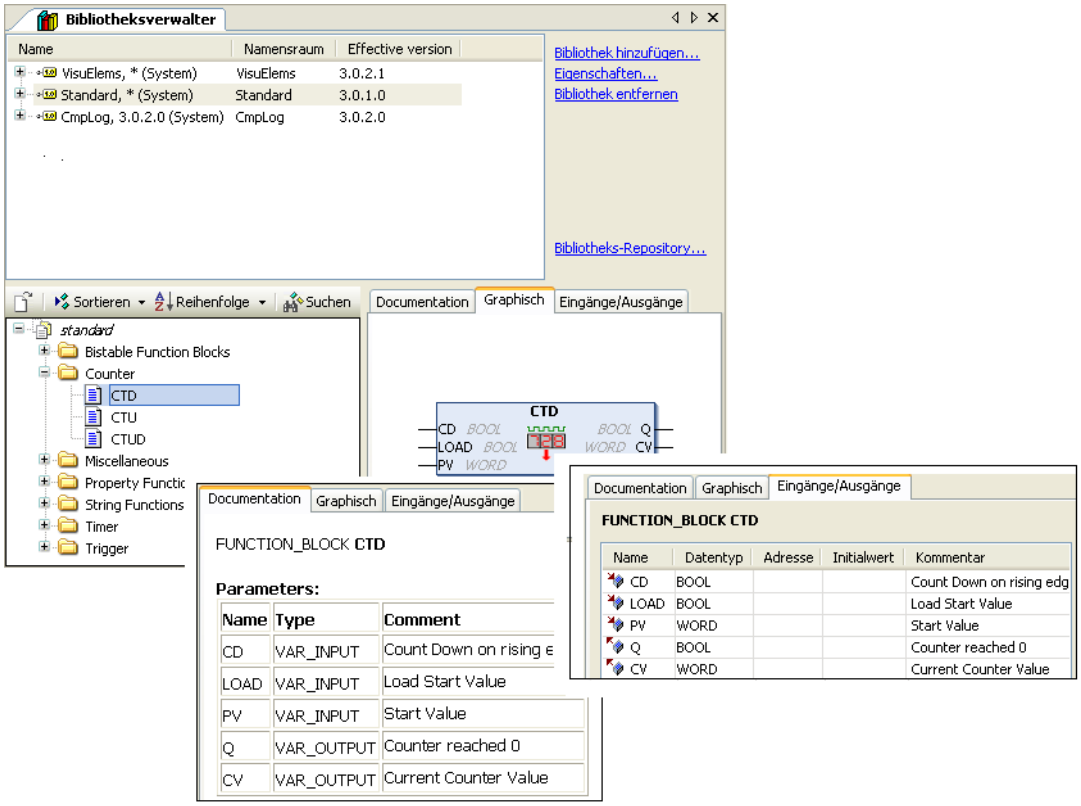


Abb. 382 Bibliotheksverwalter Editorfenster

Sehen Sie im folgenden:

- Aufbau des Editorfensters
- Schaltflächen und Befehle im Editorfenster

Generelle Informationen zur Bibliotheksverwaltung in **»PLC Designer«: siehe Bibliotheksverwaltung**.

Aufbau des Editorfensters

Der obere Teil des Bibliotheksverwalters zeigt die **aktuell im Projekt eingebundenen** Bibliotheken,. Folgende Informationen werden gegeben:

Name: Titel, Version und optional der Firmenname wie sie im Dialog 'Zusammenfassung' in den Projektinformationen während der Erstellung der Bibliothek definiert wurden.

Namensraum: Der Standard für den Namensraum einer Bibliothek ist "<library name>", **außer** es wurde explizit ein anderer Namensraum in den Projektinformationen des Bibliotheksprojekts definiert. Der Namensraum muss dem Bausteinbezeichner vorangestellt werden, wenn eindeutig auf einen Baustein zugegriffen werden soll, der mehrfach im Projekt vorhanden ist. Der Standard-Namensraum einer eingebundenen Bibliothek kann für die lokale Verwendung im Projekt auch modifiziert werden. Dies erfolgt im 'Eigenschaften'-Dialog, siehe dazu den entsprechenden Befehl, der weiter unten beschrieben ist.

Effective Version: "Effective version" gibt die Version der Bibliothek an, die gemäß der Definition in den Bibliotheks-Eigenschaften aktuell im Projekt verwendet wird.

Bibliotheken, die automatisch durch ein Plug-In im Projekt eingefügt wurden, sind in grauer, die manuell eingefügten (Bibliothek hinzufügen...) in schwarzer Schrift dargestellt.

Ein Symbol vor dem Bibliotheksnamen zeigt den Typ der Bibliothek an:

	»PLC Designer« V3.x Bibliothek (enthält Versionsinformationen).
	Die referenzierte Bibliotheksdatei konnte nicht gefunden werden oder ist keine gültige Bibliotheksdatei (siehe entsprechende Meldung in Kategorie 'Bibliotheksverwalter' im Meldungsfenster). Siehe in diesem Fall: "Bibliothek erneut laden'.

Wenn eine Bibliothek **Abhängigkeiten** von anderen Bibliotheken (referenzierte Bibliotheken) hat, werden diese - wenn sie gefunden werden - automatisch ebenfalls eingebunden und mit vorangestelltem Symbol in einem Unterbaum des Eintrags dargestellt. Ein solcher Unterbaum kann über ein Plus- oder Minuszeichen geöffnet bzw. geschlossen werden. Als Beispiel sehen Sie oben in Abbildung 'Bibliotheksverwalter Editorfenster' die Bibliothek "VisuInputs" .

Im unteren linken Teil des Editors werden die **Bausteine** der gerade ausgewählten Bibliothek ebenfalls in Baumstruktur dargestellt. Die üblichen Sortier- und Suchfunktionen stehen in einer Menüleiste bereit.

Im unteren rechten Teil finden sich folgende Registerblätter:

Dokumentation: Die Komponenten des gerade links ausgewählten Bibliotheksbausteins werden in einer Tabelle dargestellt, mit (Variablen)Name, Datentyp und dem Kommentar, der ggfs. beim Erstellen der Bibliothek der Deklaration beigefügt wurde. Somit ist das Einfügen solcher Kommentare ein einfacher Weg, dem Anwender automatisch Dokumentation eines Bausteins zur Verfügung zu stellen.

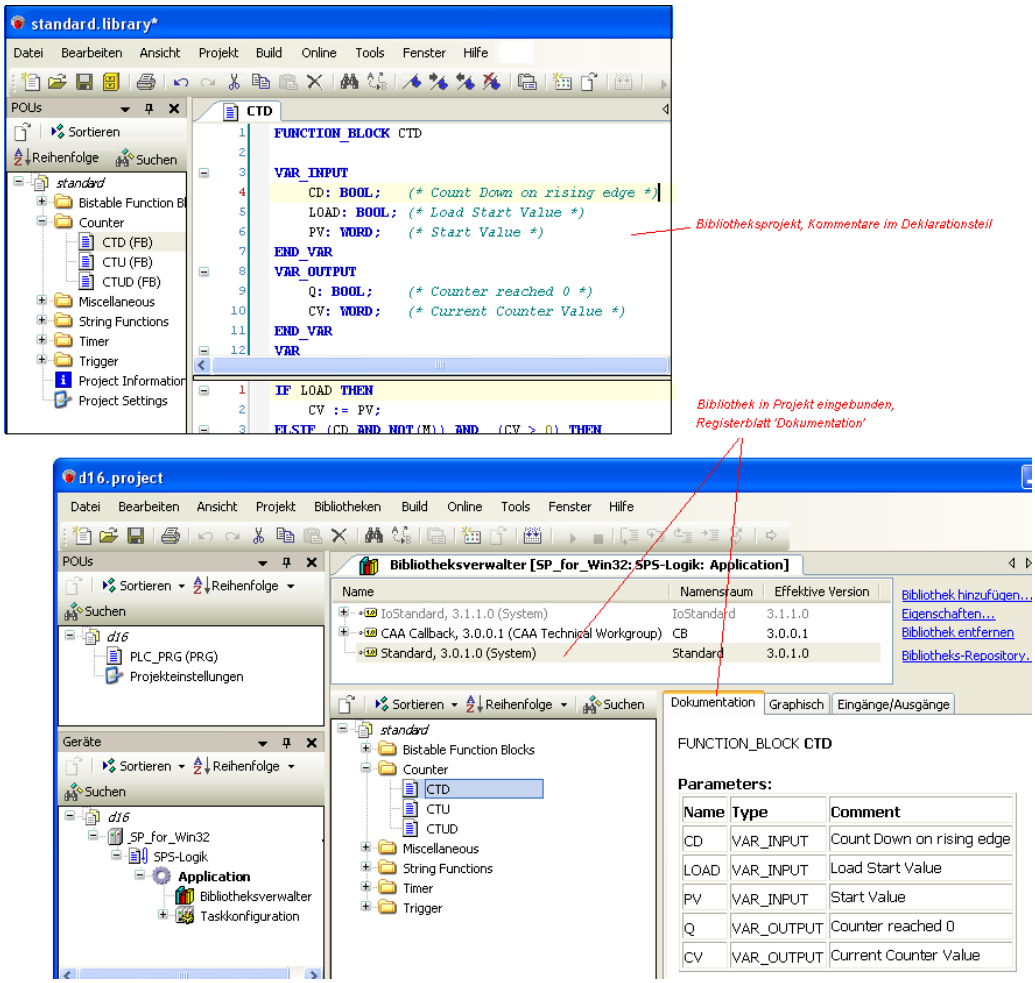


Abb. 383 Beispiel kommentierte Bibliotheksbausteine

Eingänge/Ausgänge: Die Komponenten des gerade links ausgewählten Bibliotheksbausteins werden in einer Tabelle dargestellt, mit (Variablen)Name, Datentyp, Adresse, Initialwert und Kommentar, wie in der Bibliothek definiert.

Graphisch: Graphische Darstellung des Bausteins.

Schaltflächen und Befehle im Editorfensters

Die folgenden Befehle sind im Editorfenster verfügbar, wenn einer oder mehrere Einträge im Baum der eingebundenen Bibliotheken selektiert sind. Sie sind teilweise auch im Menü 'Bibliotheken' zu finden, welches standardmäßig in der Menüleiste erscheint, solange im Bibliotheksverwalter gearbeitet wird:

Bibliothek hinzufügen... für das Einbinden einer Bibliothek ins Projekt; Voraussetzung: die Bibliothek muss auf dem System installiert sein.

Eigenschaften... für Einstellungen zum Namensraum, und - für den Fall, dass die Bibliothek später als "referenzierte Bibliothek" in einem anderen Projekt erscheint - für Einstellungen zur Versionshandhabung, Anzeige und Zugriff.

Bibliothek entfernen: Die gerade in der Liste der eingebundenen Bibliotheken selektierte wird entfernt.

Bibliotheks-Repository.... für das Definieren von Ablageorten und für die Installation oder Deinstallation von Bibliotheken.

Eine entsprechende Meldung erscheint, wenn versucht wird, eine Bibliothek einzufügen, die bereits im Projekt eingebunden ist.

13.10.2 Bibliotheksmenü

Solange im Bibliotheksverwalter gearbeitet wird, erscheint standardmäßig in der Menüleiste das Menü 'Bibliotheken' mit entsprechenden Befehlen.

Beachten Sie auch den Befehl Bibliotheks-Repository..., der standardmäßig im 'Tools'-Menü zu finden ist.

Siehe Bibliotheksverwaltung **für generelle Informationen.**

13.11 Netzwerkvariablenlisten Editor

Der NVL-Editor ist ein Deklarationseditor für das Erstellen von Netzwerkvariablenlisten. Er arbeitet gemäß den aktuellen Texteditor-Optionen und im Onlne-Modus wie allgemein für den Deklarationseditor beschrieben. Die Deklaration der Netzwerkvariablen muss mit dem Schlüsselwort "VAR_GLOBAL" beginnen und mit "END_VAR" enden. Diese Schlüsselwörter werden automatisch vorgegeben. Tragen Sie dazwischen gültige Deklarationen von globalen Variablen ein.

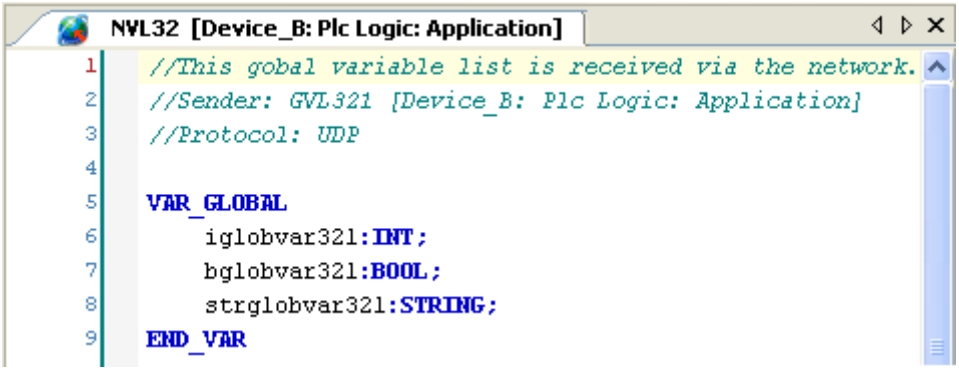


Abb. 384 NVL-Editor

13.12 Rezepturverwalter

Hinweis: Diese Funktionalität ist nur verfügbar, wenn sie vom aktuell verwendeten Feature-Set unterstützt wird.

Das Plug-In 'Recipe Manager' stellt Funktionen zur Verwaltung von benutzerdefinierten Variablenlisten, genannt "**Rezepturdefinitionen**", bereit, innerhalb denen verschiedene Werte-Sets für die Variablen definiert sind, genannt "**Rezepturen**". Solche Rezepturen können verwendet werden, um die Variablen auf der Steuerung zu setzen und zu überwachen. Dazu können sie aus der Steuerung gelesen und dorthin geschrieben werden und sie können in Dateien gespeichert und von solchen wieder geladen werden. Diese Aktionen sind mit Hilfe von Visualisierungselementen möglich, die entsprechend konfiguriert werden müssen.

Falls sich der Rezepturverwalter auf einer anderen Steuerung befindet als die von den Rezepturen betroffene Applikation selbst, so wird der Data Server benutzt, um die Rezepturen zu schreiben oder zu lesen. Das Lesen/Schreiben erfolgt hierbei synchron, d.h. alle in einer Rezeptur vorhandenen Variablen werden zum selben Zeitpunkt aktualisiert. Durch einen Aufruf von g_RecipeManager.LastError nach dem Lesen/Schreiben kann überprüft werden, ob die Übertragung geklappt hat (g_RecipeManager.LastError=0 in diesem Fall).

Der Rezepturverwalter ist ein Objekt, das im Gerätebaum unterhalb einer Applikation mit dem Befehl 'Objekt hinzufügen' eingefügt wird.

Eine oder mehrere **Rezepturdefinitionen** können unterhalb des Rezepturverwalter-Objekts ebenfalls über 'Objekt hinzufügen' eingehängt werden. Eine Rezepturdefinition mit den in ihr definierten Rezepturen wird im Editor des Rezepturverwalters in Tabellenform dargestellt und kann dort bearbeitet werden. Siehe die Hilfeseite zur Rezepturdefinition für weitere Informationen zur Handhabung von Rezepturen im Online-Betrieb.

Speicherung: Standardmäßig werden die Rezepturen automatisch gemäß den folgenden Einstellungen in Dateien gespeichert:

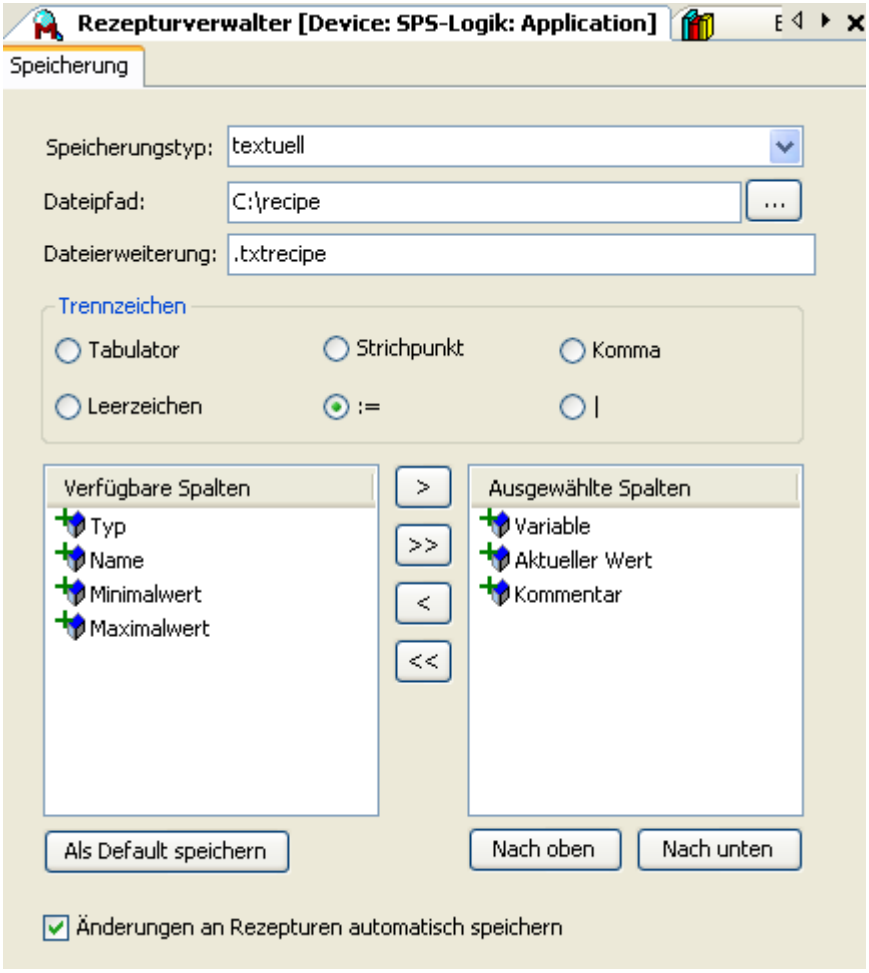


Abb. 385 Rezepturverwalter - Speicherung

Einer der angegebenen **Speicherungstyp**en (textuell oder binär) sowie der Speicherort (**Dateipfad**) und die **Dateierweiterung** der zu speichernden Rezeptur sind anzugeben.

Bei textueller Speicherung werden die zu speichernden Spalten durch ein **Trennzeichen** getrennt, für das eine der sechs Optionen ausgewählt werden kann.

Im unteren linken Teil des Dialogs (**Verfügbare Spalten**) sind alle Spalten der Rezepturdefinition mit ihrer Spaltenüberschrift aufgelistet. Im rechten Teil (**Ausgewählte Spalten**) stehen die bereits ausgewählten Spalten der Rezepturdefinition, d.h. diejenigen, die gespeichert werden sollen. Mindestens steht hier die Spalte "Aktueller Wert" und kann nicht deselektiert werden.

Die anderen Spalten können nach rechts oder links verschoben werden, indem der entsprechende Eintrag selektiert und die Schaltflächen [>] oder [<] gedrückt werden. Mit [>>] oder [<<] können alle Einträge gleichzeitig verschoben werden. Die Schaltflächen [Up] und [Down] kann dann die Reihenfolge der ausgewählten Spalten im rechten Fenster verändert werden, gemäß der die Spalten dann auch in der Speicherdatei angeordnet werden.

Für jede Rezeptur wird eine Datei **<Rezeptur>.<Rezepturdefinition>.<Dateierweiterung>** im angegebenen Verzeichnis angelegt. Die Dateien werden bei jedem Neustart der Applikation in den Rezepturverwalter geladen. Solange die Option **Änderungen an Rezepturen automatisch speichern** aktiviert ist, werden die Dateien bei jeder Rezepturänderung zur Laufzeit aktualisiert. Wenn das nicht erwünscht ist, muss die Option deaktiviert werden.

Mit **Als Default speichern** werden die im Dialog vorliegenden Einstellungen ab sofort als Standardeinstellungen für jeden weiteren im Projekt eingefügten Rezepturverwalter verwendet.

13.12.1 Rezepturdefinition

Der Rezepturverwalter verwaltet eine oder mehrere Rezepturdefinitionen. Eine "**Rezepturdefinition**" enthält eine benutzerdefinierte Liste von Variablen und eine oder mehrere "**Rezepturen**" (Werte-Sets) für diese Variablen. Durch die Verwendung verschiedener Rezepturen kann somit beispielsweise einem Variablen-Set auf der Steuerung mit einer einzigen Aktion ein anderes Set an Werten zugewiesen werden.

Rezepturdefinition

Eine Rezepturdefinition muss als Objekt unterhalb des Rezepturverwalter-Objekts im Gerätebaum hinzugefügt werden.

Wenn ein Rezepturdefinitionsobjekt selektiert ist, kann der zugehörige Editor über einen weiteren Mausklick auf das Objekt oder über Befehl 'Objekt bearbeiten' geöffnet werden.

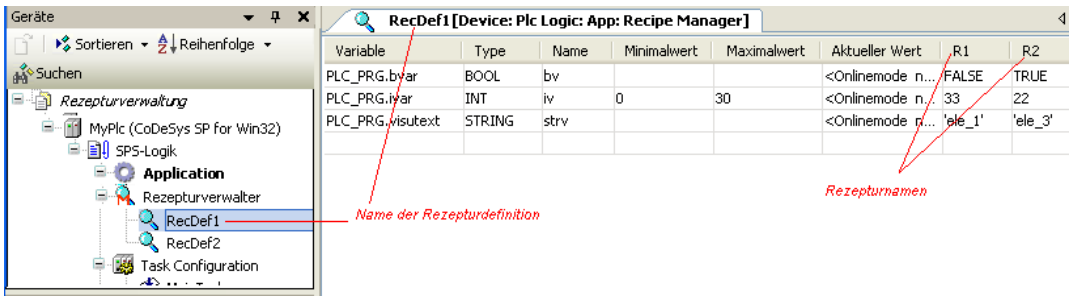


Abb. 386 Rezepturdefinition, Editorfenster

Das Editorfenster trägt in der Titelleiste den Namen der Rezepturdefinition.

In einer Tabelle können mehrere Projektvariablen eingetragen werden, für die eine oder mehrere Rezepturen erstellt werden sollen. Am Ende der Tabelle gibt es immer eine leere Zeile für das Einfügen einer weiteren Variable.

Im Feld der Spalte **Variable** geben Sie den Pfad einer Projektvariable ein, z.B. "plc_prg.ivar". In den Editiermodus gelangen Sie z.B. über einen Doppelklick auf das Feld. Dann steht auch die Eingabehilfe über Schaltfläche ... zur Verfügung. Der Daten**Typ** wird automatisch eingetragen. Optional kann ein symbolischer **Name** für die Variable hinzugefügt werden.

Optional kann der Wertebereich, der auf die Variable geschrieben werden darf, über einen **Minimalwert** und **Maximalwert** eingeschränkt werden.

Der **Aktuelle Wert** wird im Online-Modus angezeigt.

Eine Variable (Zeile) kann mit Entf (Taste bzw. Befehl) aus der Tabelle gelöscht werden, wenn eine der Zellen selektiert ist. Mehrfachselektion von Zeilen ist möglich, wenn während des Auswählens die Strg Taste gedrückt gehalten wird. Die ausgewählten Zeilen können über Copy&Paste kopiert werden. Der kopierte Inhalt wird dabei über der gerade selektierten Zeile eingefügt. Dabei werden Rezepturwerte in der entsprechende Rezepturspalte eingefügt, falls vorhanden.

Im Online-Modus kann über ein entsprechend konfiguriertes Visualisierungselement oder mit Hilfe der entsprechenden Methoden des Funktionsbausteins "RecipeManCommands" der Bibliothek Recipe_Management.library eine bestehende Rezepturdefinitionen verändert werden. Dazu gehören die nachfolgenden Methoden:

- **ReadRecipe**: Die aktuellen Variablenwerte werden in die Rezeptur übernommen.
- **WriteRecipe**: Die Rezeptur wird in die Variablen geschrieben.
- **SaveRecipe**: Die Rezeptur wird in eine standardisierte Rezeptur-Datei gespeichert.
- **LoadRecipe**: Die Rezeptur wird aus einer standardisierte Rezeptur-Datei geladen.
- **CreateRecipe**: Eine neue Rezeptur in der Rezepturdefinition wird angelegt.
- **DeleteRecipe**: Eine bestehende Rezeptur wird aus der Rezepturdefinition gelöscht.

Wird ein neuer Download durchgeführt, werden alle Rezepturen aus dem Projekt neu gespeichert. Alle anderen Rezepturen (dynamisch erzeugte Rezepturen) bleiben unverändert erhalten.

Für jede Rezeptur innerhalb der Rezepturdefinition gibt es eine Tabellenspalte, die mit dem **Rezepturnamen** betitelt ist:

Rezepturen

Eine Rezeptur kann im Offline- oder Online-Modus **eingefügt** oder **gelöscht** werden. Im Offline-Modus stehen die Befehle 'Rezeptur einfügen' bzw.'Rezepturen entfernen' im Rezepturverwalter zur Verfügung. Im Online-Modus erfolgt es entweder durch eine Eingabe auf ein entsprechend konfiguriertes **Visualisierungselement** oder mit Hilfe der entsprechenden Methoden des Funktionsbausteins "**RecipeManCommands**" der Bibliothek **Recipe_Management.library**.

Beim Einfügen einer Rezeptur wird eine weitere Spalte am rechten Ende der Tabelle eingefügt, die mit dem Namen der Rezepturdefinition betitelt ist (siehe Abbildung oben). Die Felder der Rezepturspalte können dann mit Variablenwerten gefüllt werden. Somit können für dieselben Variablen mehrere Werte-Sets vorbereitet werden.

Handhabung der Rezepturen im Online-Betrieb:

Die Handhabung der Rezepturen (anlegen, lesen, schreiben, in Datei speichern, aus Datei laden) erfolgt entweder durch die Verwendung der Methoden des Funktionsbausteins **RecipeManCommands** der Bibliothe **Recipe_Management.libray** im Applikations-Code, oder über Eingaben durch **Visualisierungselemente**.

Note: Zum Speichern der Rezepturen in Dateien, die beim Neustart der Applikation geladen werden, siehe den Rezepturverwalter-Dialog.

- Eine Beschreibung der RecipeManCommands Methoden ist in der Dokumentation innerhalb der Bibliothek enthalten.
- Zur geeigneten "Eingabe"-Konfiguration eines Visualisierungselements siehe die Hilfeseite zu Visualisierungselemente -Eigenschaften (Kategorie "Eingabe", "Befehl ausführen").

Die folgenden Methoden bezüglich einer Rezeptur sind möglich:

Rezeptur erstellen (=Rezeptur einfügen)	Eine neue Rezeptur wird in der angegebenen Rezepturdefinition angelegt.
Rezeptur lesen	Die aktuellen Werte der Variablen der angegebenen Rezepturdefinition werden von der Steuerung gelesen und in die angegebene Rezeptur geschrieben. Dabei werden die Werte implizit gespeichert (in einer Datei auf der Steuerung) und gleichzeitig in der Tabelle der Rezepturdefinition im »PLC Designer« Rezepturverwalter angezeigt. In anderen Worten: Die in »PLC Designer« verwaltete Rezeptur wird mit den Werten aus der Steuerung aktualisiert.
Rezeptur schreiben	Die Werte der angegeben Rezeptur, wie sie im Rezepturverwalter vorliegen, werden auf die entsprechenden Variablen in der Steuerung geschrieben.
Rezeptur speichern	Die Werte der gegebenen Rezeptur werden in eine Datei mit Erweiterung ***.txtrecipe** gespeichert, deren Name definiert werden muss. Dazu öffnet der Standarddialog zum Speichern einer Datei. ACHTUNG: Die implizit verwendeten Rezeptur-Dateien, die als Zwischenablage beim Lesen und Schreiben von Rezepturen nötig sind, dürfen dabei nicht überschrieben werden. D.h. der Name für die neue Datei muss anders lauten als "<Rezepturname>.<Rezepturdefinitionsname>.txtrecipe" !
Rezeptur laden	Die Rezeptur, die in einer Datei gespeichert wurde (s.o. "Rezeptur speichern"), kann aus dieser Datei wieder geladen werden. Zur Auswahl der Datei wird der Standard-Dateiauswahl-Dialog geöffnet. Der Filter ist automatisch auf Dateierweiterung "*.txtrecipe" gestellt. Nach dem Laden wird die Anzeige der betreffenden Rezeptur im »PLC Designer« Rezepturverwalter aktualisiert.
Rezeptur löschen (=Rezeptur entfernen)	Die angegebene Rezeptur wird gelöscht.
Rezeptur ändern	Die Werte der Rezeptur können geändert werden, so daß zum Beispiel bei einer nachfolgenden Rezeptur-schreiben-Aktion die zugehörigen Projektvariablen mit den neuen Werten beschrieben werden.

13.12.2 RecipeManCommands

CreateRecipe: Diese Methode erzeugt eine neue Rezeptur in der spezifischen Rezepturdefinition und anschließend werden die aktuellen PLC-Werte in die neue Rezeptur gelesen. Zum Schluss wird die Rezeptur in die Standard-Rezepturdatei gespeichert.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_ALREADY_EXIST, ERR_FAILED, ERR_OK

CreateRecipeNoSave: Diese Methode erzeugt eine neue Rezeptur in der spezifischen Rezepturdefinition und anschließend werden die aktuellen Werte in die neue Rezeptur gelesen.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_ALREADY_EXIST, ERR_FAILED, ERR_OK

DeleteRecipe: Diese Methode löscht eine Rezeptur in der Rezepturdefinition.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

DeleteRecipeFile: Diese Methode löscht die Standard-Rezepturdatei einer Rezeptur.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_OK

LoadAndWriteRecipe: Diese Methode lädt eine Rezeptur aus einer spezifischen Rezepturdatei und anschließend wird die Rezeptur in die PLC-Variablen geschrieben.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_RECIPE_MISMATCH, ERR_FAILED, ERR_OK

LoadFromAndWriteRecipe: Diese Methode lädt eine Rezeptur in die spezifische Rezepturdatei und anschließend wird die Rezeptur in die PLC-Variablen geschrieben.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

FileName - Name der Datei.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_RECIPE_MISMATCH, ERR_FAILED, ERR_OK

LoadRecipe: Diese Methode lädt eine Rezeptur aus der Standard-Rezepturdatei.

Der Name der Standard-Rezepturdatei ist <Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung>

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_RECIPE_MISMATCH, ERR_FAILED, ERR_OK

ReadAndSaveRecipe: Diese Methode liest die aktuellen PLC-Werte in die Rezeptur und anschließend wird die Rezeptur in einer Standard-Rezepturdatei gespeichert. Der Inhalt einer eventuell existierenden Datei wird überschrieben.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_SAVE_ERR, ERR_FAILED, ERR_OK

ReadAndSaveRecipeAs: Diese Methode liest die aktuellen PLC-Werte in eine Rezeptur ein und anschließend wird die Rezeptur in eine spezifische Rezepturdatei gespeichert. Der Inhalt einer eventuell existierenden Datei wird überschrieben.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

FileName - Name der Datei.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_SAVE_ERR, ERR_FAILED, ERR_OK

SaveRecipe: Diese Methode speichert die Rezeptur in der Standard-Rezepturdefinition. Der Inhalt einer eventuell existierenden Datei wird überschrieben. Der Name der Standard-Rezepturdatei ist <Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung>

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_SAVE_ERR, ERR_FAILED, ERR_OK

ReadRecipe: Diese Methode liest die aktuellen PLC-Werte in die Rezeptur ein.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

WriteRecipe: Diese Methode schreibt die Rezeptur in die PLC-Variablen.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

ReloadRecipes: Diese Methode lädt die Liste der Rezeptur von einem File-System.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_FAILED, ERR_OK

GetRecipeCount: Diese Methode gibt die Anzahl der Rezepturen der korrespondierenden Rezepturdefinition zurück.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

Rückgabewerte:

-1 : falls die Rezepturdefinition nicht gefunden wurde.

GetRecipeNames: Diese Methode gibt die Rezepturnamen der korrespondierenden Rezepturdefinition zurück.

Parameter:

RecipeDefinitionName - Name der Rezepturdefiniton.

pStrings - Zeiger auf Strings, in denen die Rezepturwerte gespeichert werden sollten.

iSize - Die Größe des String-Arrays.

iStartIndex - Der Startindex. Dieser kann bei einer Scrollingfunktion benutzt werden.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_FAILED, ERR_OK

Beispiel:

Es gibt 50 Rezepturen. Will man eine Tabelle erzeugen, die 10 Rezepturnamen gleichzeitig anzeigt, wird ein String-Array definiert:

strArr: ARRAY[0..9] OF STRING;

Korrespondierend zum iStartindex kann man dann den Rezepturnamen für einen spezifizierten Bereich bekommen.

iStartIndex := 0; Die Namen 0..9 werden zurückgegeben.

iStartIndex := 20; Die Namen 20..29 werden zurückgegeben.

In diesem Beispiel gilt:

iSize := 10;

GetRecipeValues: Diese Methode gibt die Rezepturwerte einer korrespondierenden Rezeptur zurück.

Parameter:

RecipeDefinitionName - Name der Rezepturdefiniton.

RecipeName - Name der Rezeptur.

pStrings - Zeiger auf Strings, in denen die Rezepturwerte gespeichert werden sollen.

iSize - Die Größe des String-Arrays.

iStartIndex - Der Startindex, der für eine Scrollingfunktion benutzt werden kann.

iStringLength - Die String-Länge der Strings im Array.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

Beispiel:

Es gibt 50 Rezepturen. Will man eine Tabelle erzeugen, die 10 Rezepturnamen gleichzeitig anzeigt, wird dazu ein String-Array definiert:

strArr: ARRAY[0..9] OF STRING;

Korrespondierend zum iStartindex kann man dann den Rezepturnamen für einen spezifizierten Bereich bekommen.

iStartIndex := 0; Die Namen 0..9 werden zurückgegeben.

iStartIndex := 20; Die Namen 20..29 werden zurückgegeben.

In diesem Beispiel gilt:

iStringLength := 80;

iSize := 10;

GetRecipeValueNames: Diese Methode gibt die Namen der Rezepturvariablen der korrespondierenden Rezeptur wieder.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition

RecipeName - Name der Rezeptur

pStrings - Zeiger auf Strings, in denen die Rezepturwerte gespeichert werden sollen.

iSize - Die Größe des String-Arrays.

iStartIndex - Der Startindex, der für eine Scrollingfunktion benutzt werden kann.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

Beispiel:

Es gibt 50 Rezepturen. Will man eine Tabelle erzeugen, die 10 Rezepturnamen gleichzeitig anzeigt, wird ein String-Array definiert: strArr: ARRAY[0..9] OF STRING; Korrespondierend zum iStartindex kann man dann den Rezepturnamen für einen spezifizierten Bereich bekommen.

iStartIndex := 0; Die Namen 0..9 werden zurückgegeben.

iStartIndex := 20; Die Namen 20..29 werden zurückgegeben.

In diesem Beispiel gilt:

iSize := 10;

SetRecipeValues: Diese Methode setzt die Rezepturwerte in der korrespondierenden Rezeptur.

Parameter:

RecipeDefinitionName - Name der Rezepturdefinition.

RecipeName - Name der Rezeptur.

pStrings - Zeiger auf Strings, in denen die Rezepturwerte gespeichert werden sollen.

iSize - Die Größe des String-Arrays.

iStartIndex - Der Startindex, der für eine Scrollingfunktion benutzt werden kann.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

Beispiel:

Es gibt 50 Rezepturen. Will man eine Tabelle erzeugen, die 10 Rezepturnamen gleichzeitig anzeigt, wird ein String-Array definiert: strArr: ARRAY[0..9] OF STRING; Korrespondierend zum iStartindex kann man dann den Rezepturnamen für einen spezifizierten Bereich bekommen.

iStartIndex := 0; Die Namen 0..9 werden zurückgegeben.

iStartIndex := 20; Die Namen 20..29 werden zurückgegeben.

In diesem Beispiel gilt:

iStringLength := 80;

iSize := 10;

GetLastError: Diese Methode gibt den letzten Fehler der vorherigen Operation zurück.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_OK

ResetLastError: Diese Methode setzt den letzen Fehler zurück.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_OK

SetRecipeManager: Diese Methode setzt den Rezepturmanager mit dem der Funktionsblock arbeitet.

Rückgabewerte:

ERR_NO_RECIPE_MANAGER_SET, ERR_OK.

13.12.3 Rezepturverwalter im Online-Modus

noch nicht verfügbar

13.13 AS-Editor

Der AS-Editor (SFC-Editor), der mit dem 'SFC Editor' Plug-In installiert wird, dient dem Programmieren von POUs in der IEC 61131-3 Programmiersprache Ablaufsprache (AS). Die Sprache wird ausgewählt, wenn ein neues POU-Objekt mit Hilfe des Dialogs 'Objekt hinzufügen' im Projekt angelegt wird.

Der AS-Editor ist ein **grafischer** Editor. Allgemeine Einstellungen bezüglich Verhalten und Aussehen werden im AS-Editor **Optionen**-Dialog vorgenommen.

Der AS-Editor befindet sich im unteren Teil des Fensters, das öffnet, wenn ein AS-Objekt bearbeitet wird. Im oberen Teil ist der Deklarationseditor enthalten.

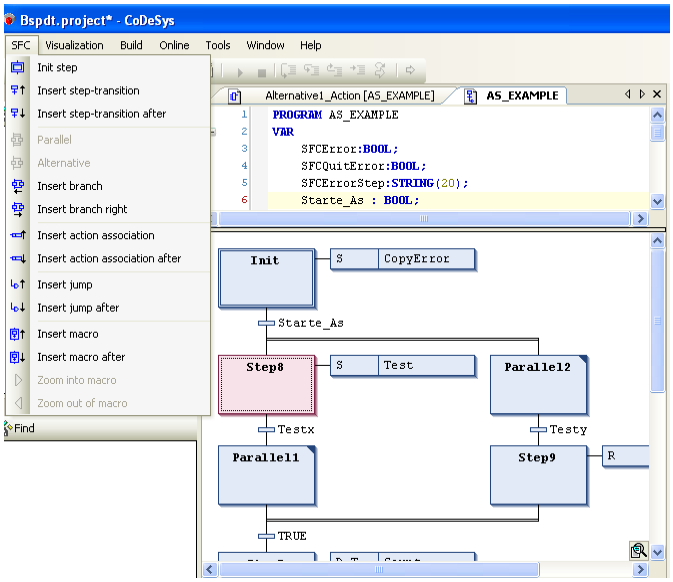


Abb. 387 AS Editor

Die Elemente, die in einem AS-Diagramm verwendet werden, sind standardmäßig über das"AS"-Menü verfügbar, sobald der AS-Editor aktiv ist. Sie müssen als eine Folge von Schritten, verbunden durch Transitionen, angeordnet werden. Parallel- und Alternativzweige sind möglich. Sehen Sie hierzu: Arbeiten im AS-Editor.

Die **Eigenschaften** von Schritten können in einem separaten Eigenschaften-Fenster bearbeitet werden. Unter anderem kann dort die minimale und maximale Aktivitätszeit eines Schrittes definiert werden.

Implizite Variablen können für das Überwachen der Abarbeitung eines AS-Diagramms verwendet werden (z. B. Schrittstatus, Zeitüberwachung, Reset etc.).

Die **Kommandos** für das Arbeiten im AS-Editor können aus dem Kontextmenü oder aus dem "AS"-Menü aufgerufen werden, welches standardmäßig in der Menüleiste erscheint, wenn der AS-Editor aktiv ist.

Vergleich zum »PLC Designer« V2.3 AS-Editor:

- Grundsätzlich ist die gleiche Funktionalität verfügbar. »PLC Designer« V2.3 AS-Projekte können importiert werden.
- Das Editieren ist einfacher geworden, dadurch, dass jedes Element einzeln angewählt und neu positioniert werden kann. Die Syntax im AS-Diagramm muss während des Programmierens noch nicht zwingend korrekt sein.
- Es gibt nur einen Schritt-Typ, der die in V2.3 verwendeten Typen "IEC-Schritt" und "3S-Schritt" vereint. Aktionen können nur noch als POU-Objekte bereitgestellt werden und werden einem Schritt immer in den Schritt-Eigenschaften zugewiesen.
- Makros können zur Strukturierung eines AS-Diagramms verwendet werden.

13.13.1 AS - Ablaufsprache

Die Ablaufsprache (AS) ist eine grafisch orientierte Sprache, die es erlaubt, die chronologische Abfolge einzelner Aktionen in einem Programm zu beschreiben. Zu diesem Zweck werden die **Aktionen**, die eigenständige, in einer der verfügbaren Sprachen geschriebene Programmier-Objekte sind, "**Schritt**"-Elementen zugeordnet. Der Ablauf der Abarbeitung der Schritte wird durch **Transitions**-Elemente gesteuert.

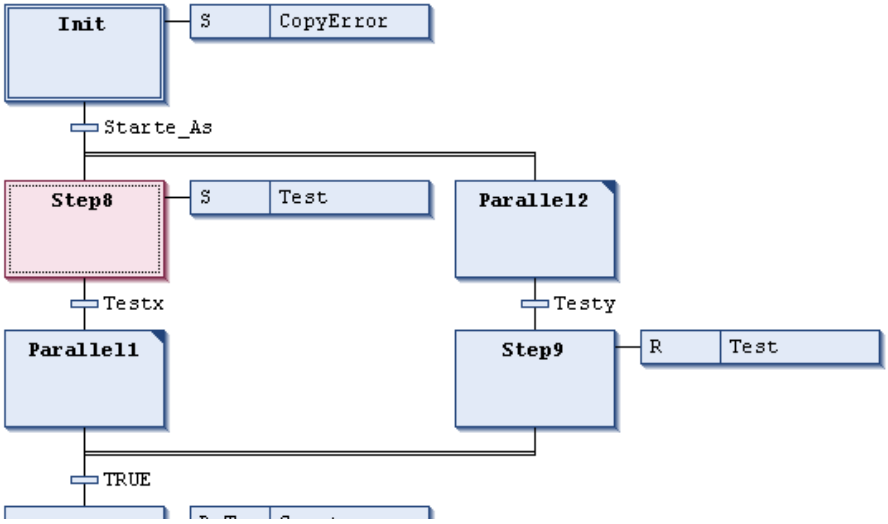


Abb. 388 Beispiel für eine Schrittabfolge in einem AS-Modul

Hinweis: Im Gegensatz zu »PLC Designer« V2.3 gibt es nur noch einen Schritt-Typ, der die Funktionalität der in »PLC Designer« V2.3 verwendeten IEC-Schritte und "3S-Schritte" vereint.

13.13.2 Cursor-Positionen in AS

Eine mögliche Cursor-Position in einem AS-Diagramm wird im AS-Editor standardmäßig durch eine **graue Schattierung** angezeigt, wenn man den Cursor über die Elemente bewegt.

Es gibt zwei Kategorien von Cursor-Positionen: Texte und Bausteinrümpfe. Sehen Sie in den folgenden Abbildungen die möglichen Positionen als grau schattiert angezeigt:

1. Texte

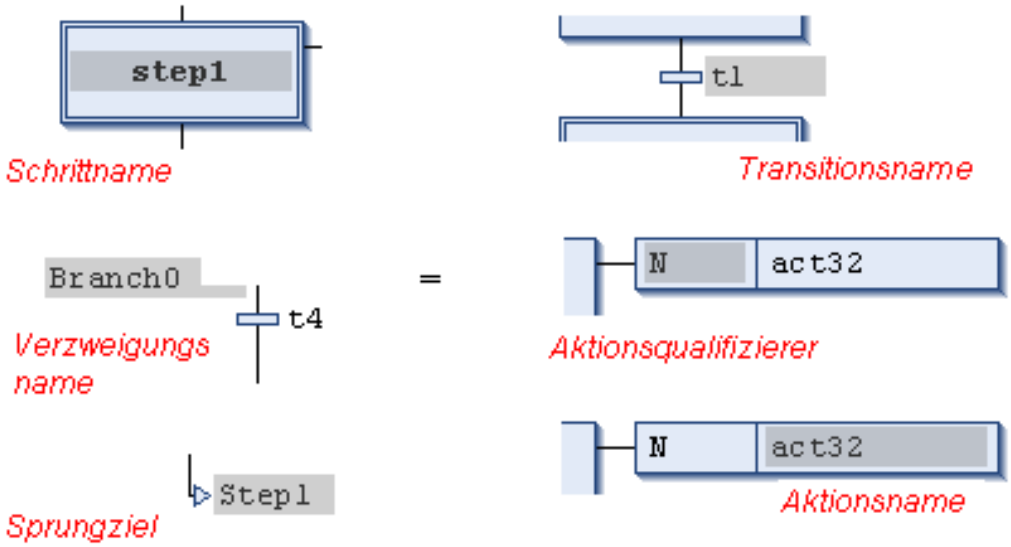


Abb. 389 Mögliche Cursor-Positionen, Texte

Wenn auf eine Text-Cursor-Position geklickt wird, wird der Text editierbar:

Abb. 390 Der Aktionsname wurde zum Bearbeiten ausgewählt

2. Elementrümpfe

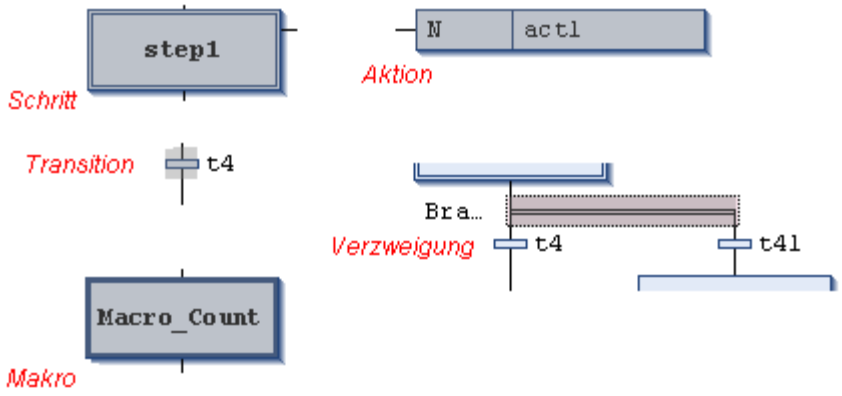


Abb. 391 Mögliche Cursor-Positionen, Elementrümpfe

Wenn man auf eine der grau schattierten Bereiche klickt, wird das **Element** ausgewählt. Es bekommt einen gepunkteten Rahmen und wird rot-schattiert dargestellt.

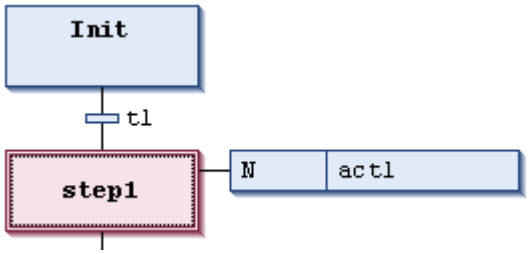


Abb. 392 Ausgewähltes Schritt-Element

13.13.3 Arbeiten im AS-Editor

Standardmäßig enthält eine neue AS-POU einen Init-Schritt und eine nachfolgende Transition. Im folgenden wird beschrieben, wie weitere Elemente eingefügt, positioniert und bearbeitet werden können:

Navigieren: Mit Hilfe der Pfeiltasten kann vom gerade ausgewählten Element zum nächsten bzw. vorherigen gesprungen werden.

Elemente einfügen

Die einzelnen AS-Elemente können über die entsprechenden Kommandos eingefügt werden, die per Default im "AS"-Menü enthalten sind. Ein Doppelklick auf ein bereits eingefügtes Schritt-, Transitions- oder Aktionselement, das keine Referenz auf ein bestehendes Projektobjekt enthält öffnet einen Dialog zum Zuweisen eines solchen.

Elemente auswählen

Ein Element oder Textbereich kann mit einem Mausklick auf eine mögliche Cursor-Position ausgewählt werden. Zum benachbarten Element kann dann mit Hilfe der Pfeiltasten gewechselt werden. Das selektierte Element wird seine Farbe nach Rot wechseln.

Im Gegensatz zum »PLC Designer« V2.3 können Schritte und Transitionen einzeln ausgewählt und neu positioniert, ausgeschnitten, kopiert, eingefügt oder gelöscht werden!

Mehrfachauswahl ist folgendermaßen möglich:

- Halten Sie die <Umschalt>-Taste gedrückt und klicken dann nacheinander auf die einzelnen Elemente, die ausgewählt werden sollen.
- Drücken Sie die linke Maustaste und ziehen ein Rechteck (gepunktete Linie) um die Elemente, die ausgewählt werden sollen.
- Verwenden Sie den Befehl Alles auswählen, der per Default im 'Bearbeiten'-Menü liegt.

Texte bearbeiten

Bei einem Mausklick auf eine Text-Cursor-Position öffnet sofort das Editierfeld, in dem der Text bearbeitet werden kann. Wenn ein Textbereich mit Hilfe der Pfeiltasten selektiert wurde, muss das Editierfeld explizit durch Drücken der <Leer>-Taste geöffnet werden.

Zugewiesene Aktionen bearbeiten

Ein Doppelklick auf die Aktionszuweisung in einem Schrittelement (Eingangs-, Schritt- oder Ausgangsaktion) oder auf ein Transitionselement mit zugewiesener Aktion öffnet die Aktion im entsprechenden Editor. Beispielsweise führen Sie einen Doppelklick auf das Dreieck aus, das in einem Schrittelement eine zugewiesene Ausgangsaktion anzeigt.

Ausschneiden, Kopieren, Einfügen von Elementen

Wählen Sie das/die Element(e) aus und verwenden Sie den jeweiligen Befehl Ausschneiden, Kopieren bzw. Einfügen (per Default aus dem 'Bearbeiten'-Menü) oder die entsprechenden Tastaturbefehle.

Beachten Sie das folgende Verhalten im Unterschied zu dem in »PLC Designer« V2.3:

- Wenn Sie ein oder mehrere Element(e) einfügen, wird der Inhalt der Zwischenablage vor der gerade angewählten Position eingefügt. Wenn gerade nichts angewählt ist, wird er am Ende des Diagramms angehängt.
- Wenn ein Zweig eingefügt wird, und das gerade angewählte Element eine Verzweigung ist, wird der eingefügte Zweig links von den bereits vorhandenen platziert.
- Wenn eine Aktion(enliste) eingefügt wird, wenn gerade ein Schritt ausgewählt ist, werden die neuen Aktionen am Beginn der Aktionenliste des Schrittes hinzugefügt bzw., wenn noch keine existiert, wird eine Aktionenliste angelegt.
- Inkompatible Elemente beim Ausschneiden/Kopieren: Wenn gleichzeitig eine Aktion(enliste) und ein Element, welches nicht der Schritt ist, zu dem die Aktion(en) gehören, ausgewählt werden, erscheint eine Meldung: "Die aktuelle Auswahl enthält unvereinbare Elemente. Es werden keine Daten in die Zwischenablage übertragen." Die Auswahl wird nicht gespeichert und kann nicht an anderer Stelle eingefügt oder kopiert werden!
- Inkompatible Elemente beim Einfügen: Wenn beim Einfügen einer Aktion(enliste) nicht gerade ein Schritt oder eine andere Aktionenlsite ist, erscheint eine Fehlermeldung: "Der Inhalt der Zwischenablage kann an der aktuellen Selektion nicht eingefügt werden." Wenn Sie versuchen, ein Element wie einen Schritt, eine Verzweigung oder eine Transition einzufügen, während gerade eine Aktionenliste angewählt ist, erscheint dieselbe Fehlermeldung.

Elemente löschen:

Wählen Sie das oder die Element(e) und verwenden Befehl Löschen oder die <Entf>-Taste. Beachten Sie folgendes:

- Löschen eines Schrittes löscht auch die zugewiesene Aktionenliste.
- Löschen des Initialschrittes macht automatisch den nachfolgenden Schritt zum Initialschritt, d.h. die Option "Initialschritt" in den Schritt-Eigenschaften wird aktiviert.
- Löschen der horizontalen Linie vor einem verzweigten Bereich löscht alle Zweige.
- Löschen aller Elemente eines Zweiges löscht den Zweig.

13.13.4 AS Element-Eigenschaften

Die Eigenschaften eines SFC-Elements können im "Eigenschaften"-Fenster angezeigt und bearbeitet werden. Dieses Fenster kann mit dem Befehl 'Eigenschaften' geöffnet werden, der standardmäßig im Menü 'Ansicht' enthalten ist.

Es hängt vom **gerade ausgewählten Element** ab, welche Eigenschaften dargestellt werden. Sie werden in Gruppen zusammengefaßt und die Gruppierungen können über die Plus- bzw. Minuszeichen auf- und zugeklappt werden.

Beachten Sie, dass Sie im 'Ansicht'-Tab des SFC-Editor Optionen-Dialogs konfigurieren können, ob eine Eigenschaft neben dem Element im AS-Diagramm angezeigt werden soll.

Sehen Sie im folgenden alle möglichen Eigenschaften:

Allgemein:

Name	Elementname, standardmäßig "<Element><fortlaufende Zahl>" , z. B. Schrittname "Schritt0", "'Schritt1", Verzweigungsname "Verzweigung0" etc.
Kommentar	Elementkommentar (Text), z. B. "Zähler Reset". Zeilenumbrüche können mit <Strg>+<Eingabe> eingefügt werden.
Symbol	Für jedes AS-Element wird eine implizite Variable angelegt, mit dem gleichen Namen wie das Element. Hier können Sie konfigurieren, ob diese Flag-Variable in die **Symbolkonfiguration** exportiert werden soll, und welches Zugriffsrecht für das Symbol in der Steuerung gelten soll. Führen Sie einen Doppelklick auf das Feld in Spalte 'Wert' aus bzw. wählen Sie das Feld an und drücken die <Leertaste>, um eine Auswahlliste zu öffnen, über die Sie eine der folgenden **Zugriffs-Optionen** setzen können: **Kein Zugriff:** Das Symbol wird in die Symbolkonfiguration exportiert, es kann aber auf der Steuerung nicht darauf zugegriffen werden. **Lesen:** Das Symbol wird in die Symbolkonfiguration exportiert und kann auf der Steuerung gelesen werden. **Schreiben:** Das Symbol wird in die Symbolkonfiguration exportiert und kann auf der Steuerung geschrieben werden. **Lesen/Schreiben:** Kombination von Lesen und Schreiben. Standardmäßig ist hier nichts (leeres Feld in der Auswahlliste) angegeben, was bedeutet dass kein Symbol in die Symbolkonfiguration exportiert wird.

Spezifisch:

Initialschritt	Diese Option ist immer nur für den Schritt aktiviert, der gerade als Initialschritt definiert ist. Standardmäßig ist dies der erste Schritt in einem AS-Diagramm. Beachten Sie: Wenn diese Eigenschaft für einen anderen Schritt aktiviert wird, muss er für den Schritt, der sie vorher hatte, **deaktiviert** werden, um Übersetzungsfehler zu vermeiden. Beachten Sie in diesem Zusammenhang auch den Befehl 'Initialschritt'.
Times:	**Hinweis**: Zeitüberschreitungen in Schritten können mit Hilfe der impliziten Variable "SFCError" überwacht werden.
Minimal aktiv	Minimale Zeitspanne, die der Schritt aktiv sein soll; mögliche Werte: Zeitangaben gemäß der IEC-Syntax (z.B. t#8s) oder TIME Variable; Default: t#0s.
Maximal aktiv	Maximale Zeitspanne, die der Schritt aktiv sein soll; mögliche Werte: Zeitangaben gemäß der IEC-Syntax (z.B. t#8s) oder TIME Variable; Default: t#0s.
Actions:	Definieren Sie hier die Aktionen, die ausgeführt werden sollen, wenn der Schritt aktiv ist. Beachten Sie die Abarbeitungsreihenfolge.
Schritt aktiviert	Diese Aktion wird ausgeführt, nachdem der Schritt aktiviert wurde ("Eingangsaktion").
Schritt aktiv	Diese Aktion wird ausgeführt, wenn der Schritt aktiv ist und eventuelle Eingangsaktion bereits abgearbeitet sind
Schritt deaktiviert	Wenn der Schritt deaktiviert wird, wird diese Aktion im darauf folgenden Zyklus ausgeführt ("Ausgangsaktion").

Hinweis: Beachten Sie die Möglichkeit, mit Hilfe der entsprechenden impliziten AS-Variablen und -Flags Informationen zum Status eines Schritts oder einer Aktion bzw. zu Zeitüberschreitungen zu erhalten.

13.13.5 AS Elemente / Toolbox

Die grafischen Elemente für das Programmieren im AS-Editor können über entsprechende Befehle eingefügt werden, die standardmäßig im "AS"-Menü bereitstehen, wenn der Editor aktiv ist.

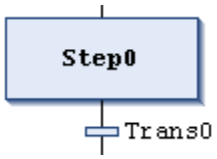
Die AS-Werkzeugbox ist noch nicht implementiert.

Sehen Sie ggfs. außerdem die Hilfeseite zum Arbeiten im AS-Editor.

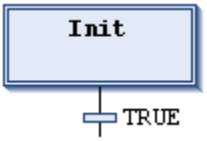
Die folgenden Elemente stehen zur Verfügung und sind nachfolgend beschrieben: Schritt, Transition, Aktion, Verzweigung (Alternativ), Verzweigung (Parallel), Sprung, Makro:

Schritt-Transition

Beispiel für Schritt und nachfolgende Transition:



Beispiel für Initialschritt und nachfolgende Transition:



Ein **Schritt** wird durch eine Box repräsentiert, die den Schrittnamen enthält und die mit der nachfolgenden Transition über eine Linie verbunden ist.

Hinweis: Schrittnamen müssen innerhalb des Gültigkeitsbereichs des "Vater"-Bausteins eindeutig sein. Bedenken Sie dies insbesondere, wenn Aktionen verwendet werden, die ebenfalls in AS programmiert sind.

Der Box-Rahmen des **Initialschrittes** besteht aus einer Doppellinie.

Beachten Sie, dass ein Schritt mit dem Befehl 'Initialschritt' oder über Setzen der entsprechenden Schritt-Eigenschaft zum **Initialschritt** gemacht werden kann.

Der **Schrittname** kann direkt, also nach Anwählen in der Box, editiert werden.

Jeder Schritt wird durch die Schritt-Eigenschaften definiert.

Hinweis: Im Gegensatz zu »PLC Designer« V2.3 gibt es nur einen Schritt-Typen (IEC-konform).

Die Aktionen, die ausgeführt werden sollen, wenn der Schritt aktiv ist, werden dem Schritt über den Befehl "Aktionsassoziation hinzufügen" bzw. in seinen Eigenschaften hinzugefügt (s.u., Aktion).

Schritte und Transitionen werden zunächst immer als Kombination **eingefügt**, mit dem Befehl 'Schritt-Transition (danach) einfügen'.

Eine **Transition** wird durch eine kleine rechteckige Box repräsentiert, die mit dem vorangehenden bzw. nachfolgenden Schritt über eine Linie verbunden ist. Sie enthält die Bedingung unter der - also sobald sie TRUE liefert - der nachfolgende Schritt aktiv wird.

Der Transitionsname bzw. die Transitionsbedingung wird rechts von dieser Box dargestellt.

Standardmäßig wird beim Einfügen der Transitionsname "**trans<n>**" vergeben, wobei n eine fortlaufende Zahl ist.

Dieser Default-Name kann nach Anwählen abgeändert werden. Dabei sind folgende Eingaben möglich:

- Name eines **Transitionsobjekts**, das im **POUs**-Fenster vorhanden ist (siehe links, Beispiel "t1"; dies erlaubt die **Mehrfachverwendung** von Transitionen!)
- Gültiger **Bedingungsausdruck** (z.B." a AND b", "TRUE"

Hinweis: Transitionen, die aus einem Transitions- oder Eigenschaften-Objekt bestehen, werden durch ein kleines Dreieck in der rechten oberen Ecke des Transitionsrechtecks gekennzeichnet.

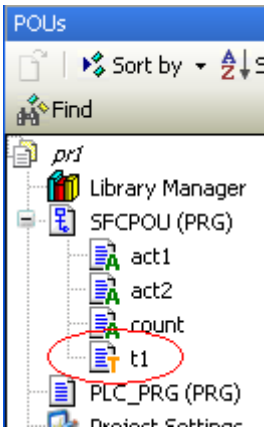


Abb. 393 Beispiel für Transition im POUs Fenster

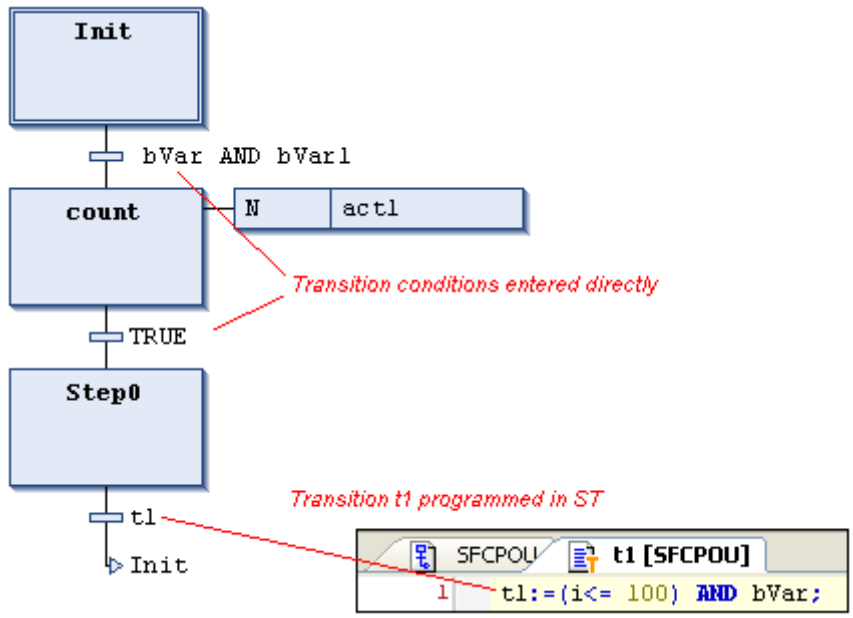


Abb. 394 Beispiele von Transitionen

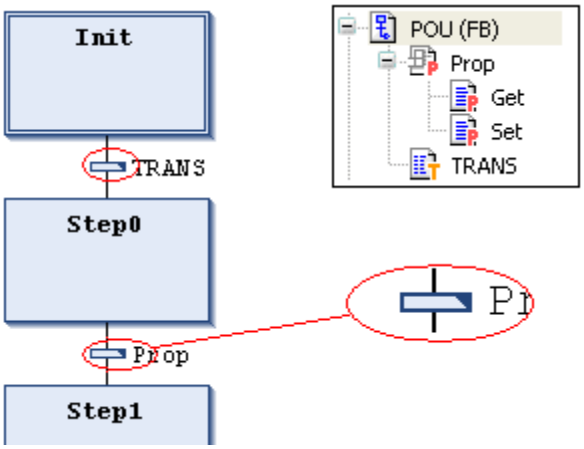


Abb. 395 Transitions- oder Eigenschaftenobjekt gekennzeichnet mit Dreieck

Eine Transitions**bedingung** muss den Wert TRUE oder FALSE haben. Somit kann sie aus einer boolschen Variable, Adresse oder Konstante, oder aus einer Reihe von Ausdrücken mit boolschem Ergebnis bestehen. Aber eine Transition darf keine Programme, Funktionsblöcke oder Zuweisungen enthalten!

Im Gegensatz zu »PLC Designer« V2.3...

... wird eine Transitionsbedingung wie eine Methodenaufruf behandelt. Die Eingabe erfolgt nach

Syntax:

```
<Transitionsname>:=<Transitionsbedingung>
```

(beispielsweise" trans1:= a=100")

oder nur

```
<Transitionsbedingung>
```

(beispielsweise "a=100")

Sehen Sie oben ein Beispiel (t1).

Im **Online-Modus** wird der nachfolgende Schritt nur aktiv, wenn die vorangehende Transition TRUE ist.

Aktion

Eine Aktion kann eine Reihe von **Anweisungen** enthalten, die in einer der gültigen Programmiersprachen geschrieben ist. Sie wird einem Schritt zugewiesen und wird im **Online-Modus** gemäß der definierten **Abarbeitungsreihenfolge** ausgeführt.

Alle Aktionen, die in AS-Schritten verwendet werden, müssen als **POUs** im Projekt bzw. unterhalb des AS-Objekts angelegt sein ( ). Mit dem Befehl 'Objekt hinzufügen' kann eine Aktion unterhalb einer AS-POU angelegt werden.

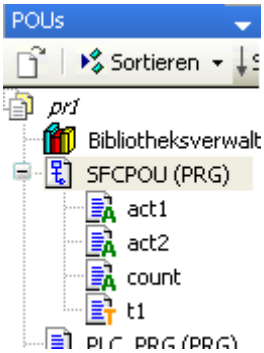


Abb. 396 Aktionen im POUs-Fenster

Hinweis: Schrittnamen müssen innerhalb des Gültigkeitsbereichs des "Vater"-Bausteins eindeutig sein. Eine in AS geschriebene Aktion darf keinen Schritt enthalten, der den gleichen Namen hat wie der Schritt, dem die Aktion zugewiesen ist.

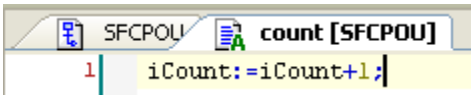


Abb. 397 Beispiel einer in ST geschriebenen Aktion

Es gibt die folgenden Typen von Aktionen:

1. IEC-konforme Schrittaktionen ("IEC-Aktion"):
Dies sind Aktionen gemäß der Norm IEC1131-3, die entsprechend ihres Qualifiers ausgeführt werden, wenn der Schritt aktiv wird und ein zweites Mal, wenn er deaktiviert wird. Wenn einem Schritt mehrere Aktionen zugewiesen sind, wird die **Aktionenliste** von oben nach unten abgearbeitet.

Für IEC-Aktionen können im Gegensatz zu "einfachen" Schrittaktionen unterschiedliche Qualifizierer verwendet werden.

Ein weiterer Unterschied zu den "einfachen" Schrittaktionen ist, dass jede IEC-Schrittaktion mit einem **Kontroll-**Flag versehen ist, das bewirkt, dass die Aktion - auch wenn sie gleichzeitig durch von einem weiteren Schritt aufgerufen wird - nur einmal zur selben Zeit ausgeführt wird. Dies kann für die einfachen Schrittaktionen nicht garantiert werden.

Eine IEC-Schrittaktion wird durch eine **zweigeteilte Box** rechts vom Schritt und mit dieser über eine Linie verbunden präsentiert. Im linken Teil zeigt sie den Aktions-**Qualifier**, im rechten den **Aktionsnamen**. Beide können direkt editiert werden.

IEC-Schrittaktionen werden mit dem Befehl 'Aktionsassoziation (danach) einfügen" **einem Schritt zugewiesen**. Einem Schritt können eine oder mehrere Aktionen zugewiesen werden. Die Position der neu eingefügte Aktion hängt von der aktuellen Cursor-Position und dem Befehl ab. Die Aktionen müssen als Objekte im Projekt vorliegen und müssen mit einem eindeutigen Aktionsnamen angegeben werden (z.B. plc_prg.a1).

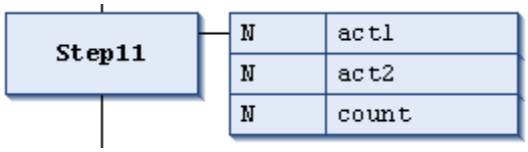


Abb. 398 IEC-konformer Schritt mit Aktionenliste

Jede Aktionenbox enthält in der ersten Spalte den Qualifier und in der zweiten den Aktionsnamen.

2. IEC-erweiternde Schrittaktionen:

Dies sind Aktionen, die in Erweiterung der des IEC-Standards verwendet werden können (in »PLC Designer« V2.3 als "3S Schrittaktionen" bekannt):

- **"Schritt aktiviert"-Aktion ("Eingangsaktion")**:

Eine Eingangsaktion wird ausgeführt, nachdem der Schritt aktiviert wurde und bevor die "Schrittaktion" ausgeführt wird.

Die Aktion wird einem Schritt über einen Eintrag im "Schritt aktiviert" Feld des Eigenschaften-Dialogs hinzugefügt. Sie wird durch ein "E" in der unteren linken Ecke der Schritt-Box angezeigt.

- "Schritt aktiv"-Aktion ("Schrittaktion"):

Eine Schrittaktion wird ausgeführt, nachdem der Schritt aktiviert wurde und eine eventuelle Eingangsaktion bereits ausgeführt ist. Allerdings wird sie im Gegensatz zu einer IEC-Schrittaktion (s.o.) kein zweites Mal ausgeführt, wenn der Schritt wieder deaktiviert wird, und kann auch nicht mit Qualifizierern versehen werden.

Die Aktion wird dem Schritt über einen Eintrag im "Schritt aktiv" Feld des Eigenschaften-Dialogs hinzugefügt. Sie wird durch ein rotes gefülltes Dreieck in der rechten oberen Ecke der Schritt-Box angezeigt.

- **"Schritt deaktiviert"-Aktion (Ausgangsaktion)**:

Eine Ausgangsaktion wird einmal ausgeführt, wenn der Schritt deaktiviert wurde. Beachten Sie allerdings, dass die Ausführung nicht mehr im gleichen Zyklus, sondern zu Beginn des nächsten erfolgt!

Die Aktion wird einem Schritt über einen Eintrag im "Schritt aktiviert" Feld des Eigenschaften-Dialogs hinzugefügt. Sie wird durch ein "X" in der unteren linken Ecke der Schritt-Box angezeigt.

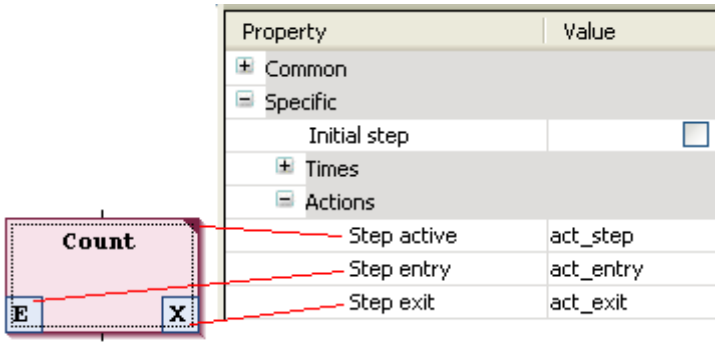
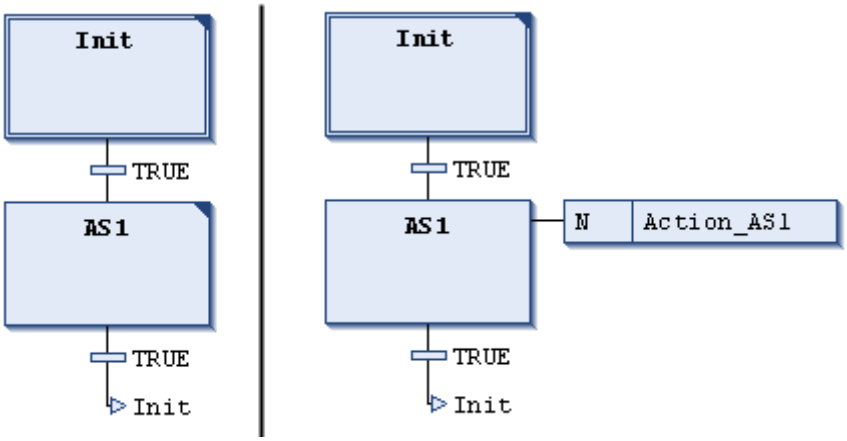


Abb. 399 IEC-erweiternde Schrittaktionen

Die Aktionen "Schritt aktiv", "Schritt aktiviert", "Schritt deaktiviert" werden in den Element-Eigenschaften definiert:

Beispiel: Unterschied IEC-konforme/erweiternde Schrittaktionen

Der wesentliche Unterschied zwischen Schrittaktionen und IEC-Aktionen mit Qualifizierer N besteht darin, dass die IEC-Aktion zweimal ausgeführt wird: ein erstes mal, wenn der Schritt aktiv ist und ein zweites Mal, wenn er deaktiviert wird. Das soll folgendes Beispiel noch einmal genauer erläutern:



Dem Schritt AS1 sei die Aktion Action_AS1 einmal als Schrittaktion (links), einmal als IEC-Aktion mit Qualifizierer N beigefügt. Da in beiden Fällen 2 Transitionen geschaltet werden, dauert es jeweils 2 SPS-Zyklen bis erneut der Initialisierungsschritt erreicht wird. Nehmen wir einmal an, in der Aktion Action_AS1 werde eine mit 0 initialisierte Zählvariable iCounter hochgezählt. Nach der erneuten Aktivierung des Schritts Init hat iCounter im linken Beispiel den Wert 1. Im rechten Beispiel hat sie jedoch den Wert 2, da die IEC-Aktion ein zweites mal aufgrund der Deaktivierung von AS1 ausgeführt wird.

Verzweigungen

Ein AS-Diagramm kann **Verzweigungen** enthalten, d.h. die Ausführungslinie kann sich in zwei oder mehrere Arme aufspalten. "Parallele Verzweigungen" werden gleichzeitig ausgeführt, bei "alternativen Verzweigungen" wird immer nur einer der Arme ausgeführt, abhängig von der vorausgehenden Transitionsbedingung. Jede Verzweigung innerhalb eines Diagramms wird durch eine horizontale einfache (alternativ) bzw. doppelte (parallel) Linie begonnen und mit durch eine solche oder einen Sprung beendet.

Parallel Verzweigung

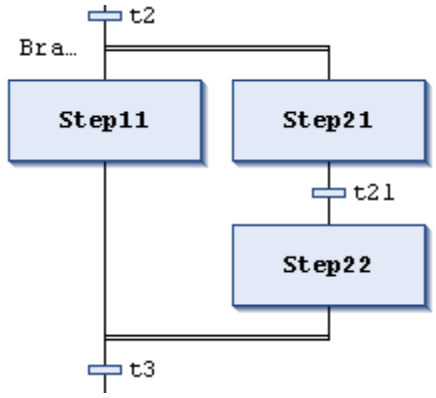


Abb. 400 Beispiel Parallelverzweigung

Die horizontale Linie vor und nach der Verzweigung ist eine **doppelte Linie**.

In einer **parallelen Verzweigung** müssen die Arme mit Schritten beginnen und enden. Parallele Arme können weitere Verzweigungen beinhalten.

Abarbeitung **im Online-Modus**: Wenn die vorausgehende Transition (t2 im links dargestellten Beispiel) TRUE ist, werden die ersten Schritte in allen parallelen Armen aktiv (Step11 und Step21). Die einzelnen Arme werden parallel abgearbeitet und erst dann die nachfolgende Transition ausgewertet (t3).

Eine parallele Verzweigung wird mit dem Befehl Verzweigung (rechts) einfügen' **eingefügt**, wenn gerade ein Schritt selektiert ist.

Beachten Sie, dass parallele und alternative Verzweigungen ineinander **übergeführt** werden können, mit Hilfe der Befehle Parallel bzw. Alternativ. Dies kann während des Aufbaus eines Programms hilfreich sein.

Der horizontalen Linie, die den Beginn einer Verzweigung bildet, wird automatisch eine **Sprungmarke** hinzugefügt, die "Branch<n>" heißt, wobei n eine fortlaufende Nummer, beginnend mit "0" ist. Diese Marke kann als Sprungziel angegeben werden..

Alternative Verzweigung

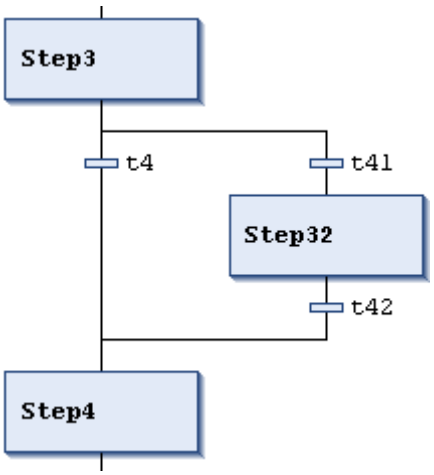


Abb. 401 Beispiel Alternative Verzweigung

Die horizontale Linie vor und nach der Verzweigung ist eine **einfache Linie**.

In einer **alternativen Verzweigung** müssen die Arme mit Transitionen beginnen und enden. Die Arme können weitere Verzweigungen beinhalten.

Wenn der Schritt, der der Verzweigung vorangeht, aktiv ist, wird jeweils die erste Transition der alternativen Arme von links nach rechts ausgewertet. Bei der ersten Transition, die TRUE liefert, wird der zugehörige Arm "geöffnet", d.h. der der Transition folgende Schritt aktiviert.

Alternative Verzweigungen werden mit dem Befehl 'Verzweigung (rechts) einfügen' eingefügt, wenn gerade eine Transition ausgewählt ist.

Beachten Sie, dass parallele und alternative Verzweigungen ineinander **übergeführt** werden können, mit Hilfe der Befehle Parallel bzw. Alternativ. Dies kann während des Aufbaus eines Programms hilfreich sein.

Sprung

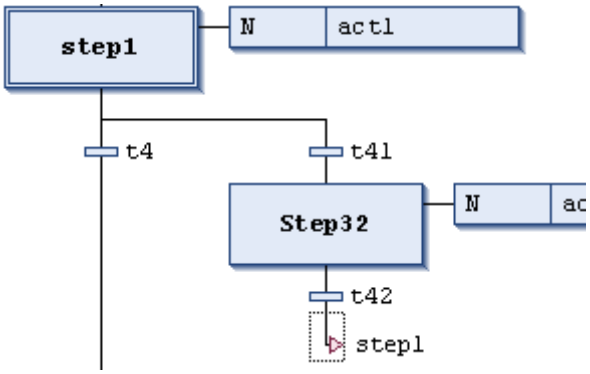


Abb. 402 Beispiel Sprung

Ein **Sprung** wird durch eine vertikale Verbindungslinie mit einem horizontalen Pfeil und dem Namen des Sprungziels dargestellt.

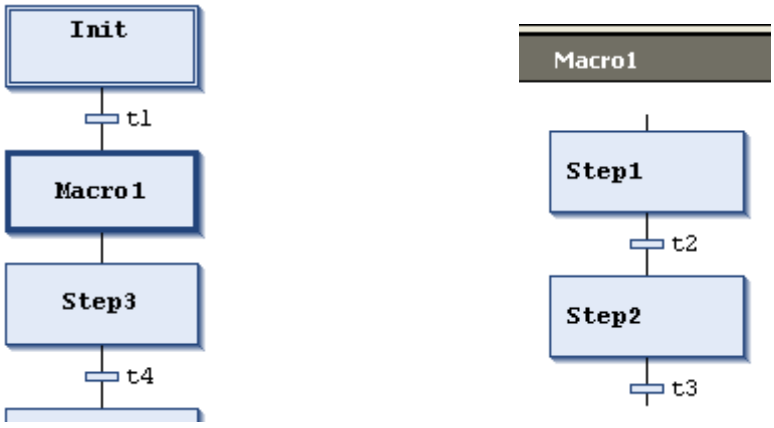
Ein Sprung definiert, welcher Schritt als nächstes ausgeführt werden soll, sobald die dem Sprung vorausgehende Transition TRUE wird. Sprünge können nötig sein, weil die Ausführungslinien sich nicht kreuzen und nicht nach oben führen können.

Neben dem obligatorischen Sprung am Ende des Diagramms können Sprünge nur am Ende von Verzweigungen **eingefügt** werden (Befehl 'Sprung (danach) einfügen' wenn gerade die letzte Transition eines Arms ausgewählt ist.

Das **Ziel** eines Sprungs wird durch den hinzugefügten Textstring definiert, der direkt editiert werden kann. Es kann ein Schrittname oder die Marke einer Parallelverzweigung sein.

Makro

Hauptansicht im SFC-Editor: Ansicht im Makro-Editor für Macro1:



Ein **Makro** wird durch eine dick eingerahmte Box dargestellt, die den Makronamen enthält.

Ein Makro enthält einen Teil des AS-Diagramms, der in der Hauptansicht des Editors nicht ausführlich dargestellt wird.

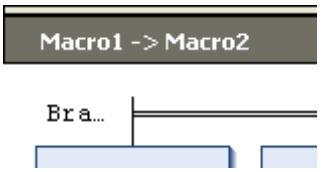
Der Abarbeitungsfluß wird durch die Verwendung von Makros nicht beeinflußt, sie dienen nur dazu, bestimmte Teile des Diagramms auszublenden, z.B. um das die Übersichtlichkeit zu verbessern.

Eine Makro-Box wird mit dem Befehl 'Makro (danach) einfügen' **eingefügt**'. Der Makroname kann direkt editiert werden.

Um den **Makro-Editor** zu öffnen, führen Sie einen Doppelklick auf die Makro-Box aus oder verwenden Sie den Befehl 'Makro anzeigen'. Hier kann genauso programmiert werden wie in der Hauptansicht des AS-Editors und der gewünschte Teil des Diagramms erstellt werden. Um den Makro-Editor wieder zu schließen, verwenden Sie Befehl 'Makro verlassen'.

Makros können wiederum Makros enthalten. Die Titelzeile des Makro-Editors zeigt immer, welchen Pfad das gerade geöffneten Makro innerhalb des Diagramms hat.

Beispiel:



13.13.6 Qualifizierer für Aktionen in AS

Um zu konfigurieren, in welcher Art eine Aktion, die einem IEC-Schritt zugewiesen ist, ausgeführt werden soll, können verschiedene Qualifizierer verwendet werden, die im Qualifizierer"-Feld eines Aktions-Elements eingetragen werden können.

Die Qualifizierer werden vom SFCActionControl Funktionsblock der **IecSfc.library** verarbeitet, die normalerweise automatisch durch die SFC-Plug-Ins im Projekt eingebunden wird.

Verfügbare Qualifizierer :

N	Non-stored	Die Aktion ist solange aktiv wie der Schritt.
R	overriding Reset	Die Aktion wird deaktiviert.
S	Set (Stored)	Die Aktion wird ausgeführt sobald der Schritt aktiv wird und wird weiter ausgeführt, auch wenn der Schritt schon deaktiviert wurde, bis sie einen Reset erhält.
L	time Limited	Die Aktion wird ausgeführt sobald der Schritt aktiv wird und wird mindestens über die gegebene Zeitspanne, höchstens aber solange bis der Schritt deaktiviert wird, ausgeführt.
D	time Delayed	Die Ausführung der Aktion startet erst, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist und der Schritt immer noch aktiv ist. Die Aktion wird ausgeführt, bis der Schritt deaktiviert wird.
P	Pulse	Die Aktion wird genau einmal ausgeführt, sobald der Schritt aktiv wird.
SD	Stored and time Delayed	Die Ausführung der Aktion startet, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist. Sie wird solange ausgeführt, bis sie einen Reset erhält.
DS	Delayed and Stored	Die Ausführung der Aktion startet erst, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist und der Schritt immer noch aktiv ist. Die Aktion wird dann ausgeführt, bis sie einen Reset erhält.
SL	Stored and time limited	Die Aktion wird ausgeführt, sobald der Schritt aktiviert wird. Sie wird solange ausgeführt, bis die gegebene Zeit abgelaufen ist oder sie einen Reset erhält.

Die Zeitangaben bei den Qualifizierern L, D, SD, DS und SL müssen im Format einer TIME Konstante erfolgen.

Hinweis: Wenn eine IEC-Aktion deaktiviert wird, wird sie ein weiteres Mal ausgeführt. Dies bedeutet, dass eine solche Aktion mindestens zweimal ausgeführt wird.

13.13.7 Implizite Variablen - AS Flags

Jedes AS-Objekt stellt implizite Variablen zur Kontrolle des Status von Schritten und IEC-Aktionen zur Laufzeit bereit. Außerdem können Variablen definiert werden, um die Abarbeitung eines AS-Diagramms zu beeinflussen und zu kontrollieren (Zeitüberschreitungen, Reset, Tip-Modus). Diese Variablen können auch implizit vom AS-Objekt erzeugt werden.

Grundsätzlich wird für jeden Schritt und jede IEC-Aktion eine implizite Variable angelegt: Eine Strukturinstanz, genauso benannt wie das Element, z.B. "step1" für einen Schritt mit Schrittnamen "step1". Beachten Sie die Möglichkeit, in den Element-Eigenschaften anzugeben, ob für dieses Flag eine Symboldefinition in die Symbolkonfiguration exportiert werden soll, und wie auf das Symbol in der Steuerung zugegriffen werden können soll.

Die Datentypen für diese impliziten Variablen sind in der Bibliothek **IecSFC.library** definiert. Diese Bibliothek wird automatisch im Projekt eingebunden, sobald ein AS-Objekt angelegt wird.

Schritt- und Aktionsstatus

Für jeden Schritt und jede IEC-Aktion wird eine implizite Variable des Typs **SFCStepType** bzw. **SFCActionType** angelegt. Die Strukturkomponenten ("flags") beschreiben den Status eines Schritts bzw. einer Aktion oder die aktuell bereits abgelaufene Zeit in einem aktiven Schritt.

Die Syntax für die implizit durchgeführte Variablendeklaration:

<stepname>: SFCStepType; bzw. _<actionname>:SFCActionType;

Hinweis: Im Unterschied zu V2.3 werden implizite Variablen für Aktionen immer mit Unterstrich geschrieben.

Folgende boolsche Flags sind für Schritt- oder Aktionsstati verfügbar:

Für Schritte:

<Schrittname>.x zeigt den Aktivationsstatus im aktuellen Zyklus.

<Schrittname>._x: zeigt den Aktivationsstatus für den nächsten Zyklus.

Wenn <Schrittname>.x = TRUE, wird der Schritt im aktuellen Zyklus ausgeführt.

Wenn <Schrittname>._x = TRUE und <Schrittname>.x = FALSE, wird der Schritt im nächsten Zyklus ausgeführt, d.h. <Schrittname>._x wird zu Beginn eines Zyklus in <Schrittname>.x kopiert.

Für Aktionen:

_<Aktionsname>.x ist TRUE, wenn die Aktion ausgeführt wird

_<Aktionsname>.x ist TRUE, wenn die Aktion aktiv ist

Symbolerzeugung:

In den Elementeigenschaften eines Schritts oder einer Aktion kann definiert werden, ob für das Schritt- bzw. Aktionsnamen-Flag eine **Symbol**definition zur eventuell erzeugten und auf die Steuerung geladenen Symbolapplikation hinzugefügt werden soll. Dazu muss in der Ansicht 'Eigenschaften' für den Schritt oder die Akton das gewünschte Zugriffsrecht in Spalte "Symbol" aktiviert werden.

Achtung: Die oben beschriebenen Flags könnten benutzt werden, um einen bestimmten Statuswert für einen Schritt zu erzwingen (forcen), d. h. um einen Schritt aktiv zu setzen. Beachten Sie jedoch, dass dadurch ein unkontrollierter Status des AS herbeigeführt wird!

Zeitabfrage über TIME-Variablen:

Das Flag "t" liefert die aktuelle Zeitspanne, die seit Aktivwerden des Schrittes verstrichen ist. Dies gilt nur für Schritte, unabhängig davon ob in den Schritteigenschaften eine Minimalzeit definiert ist oder nicht (siehe auch unten: SFCError)

Für Schritte: **<Schrittname>.t** (<Schrittname>._t ist nicht extern verwendbar)

Für Aktionen: Die impliziten Zeitvariablen werden nicht verwendet.

Kontrolle der Ausführung eines AS-Diagramms (Zeitüberschreitungen, Reset, Tip-Modus)

Implizit erzeugte Variablen mit bestimmten Namen (AS-Flags) können verwendet werden, um die Abarbeitung eines AS-Diagramms zu kontrollieren, z.B. für die Anzeige von Zeitüberschreitungen oder für das Anschalten des Tip-Modus, damit Transitionen gezielt geschaltet werden können.

Um diese Variablen verwenden zu können, müssen sie **deklariert und aktivert** werden. Dies **muss im** AS-Einstellungen Dialog erfolgen, der ein Unterdialog des Objekteigenschaften-Dialogs ist. Händische Deklaration, wie es in V2.3 nötig war, ist nur noch erforderlich um Schreibzugriff von einem anderen Baustein aus zu ermöglichen (siehe unten, Zugriff auf implizite Variablen). In diesem Fall ist folgendes zu beachten: Wenn das Flag **global** deklariert wird, muss die "Deklarieren" Einstellung im AS-Einstellungen-Dialog deaktiviert sein, damit nicht implizit eine lokale Flag-Variable deklariert wird, die dann anstelle der globalen verwendet würde! Bedenken Sie immer, dass die AS-Einstellungen eines AS-Bausteins zunächst durch die aktuellen Definitionen im AS Opfionen Dialog vorgegeben werden.

Beachten Sie, dass die Deklaration einer Flag-Variable, die nur im AS Einstellungen Dialog vorliegt, **nur in der Online-Ansicht** des AS-Bausteins **sichtbar** wird !

Beispiel der Anwendung von SFCError:

Es gibt einen AS-Baustein namens "sfc1", der einen Schritt "s2" enthält. In den Schrittattributendes Schritts sind Zeitbegrenzungen definiert. Siehe diese unten in Abbildung "Online-Ansicht von AS-Baustein sfc1".

Wenn aus irgendwelchen Gründen Schritt s2 länger aktiv bleibt als in seinen Zeiteigenschaften erlaubt ist (Zeitüberschreitung), wird ein AS Flag gesetzt, auf das die Applikation zugreifen kann.

Um diesen Zugriff zu erlauben, muss das Flag in den AS Einstellungen deklariert werden. Dazu selektieren Sie sfc1 im Geräte- bzw POUs-Fenster und wählen Befehl 'Eigenschaften' vom Kontextmenü. Öffnen Sie den Unterdialog 'AS Einstellungen' und setzen auf Registerblatt 'Variablen' für die Variable "SFCError" jeweils einen Haken in Spalte 'Deklaration' und "Aktiv". (Wenn Sie nur deklarieren, würde die Variable zwar in der Online Ansicht von sfc1 im Deklarationsteil angezeigt, wäre aber ohne Funktion.)

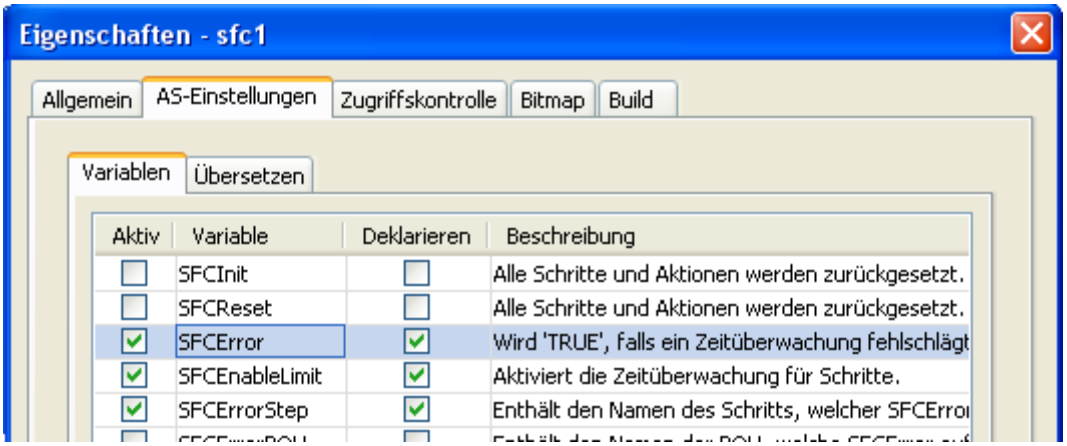


Abb. 403 AS Einstellungen

Nun können Sie SFCError innerhalb des AS, z. B. in einer Aktion, mit "SFCError" oder von außerhalb des Bausteins mit "sfc1.SFCError" ansprechen.

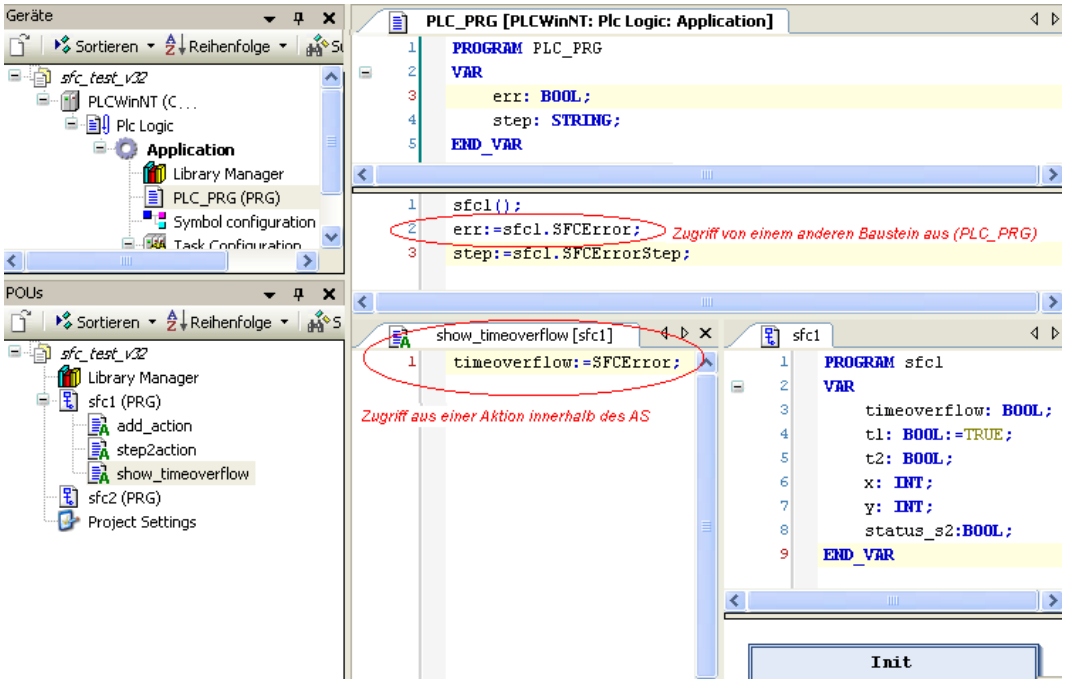


Abb. 404 Zugriff auf SFCError

SFCError wird TRUE, sobald eine Zeitüberschreitung innerhalb von sfc2 auftritt.

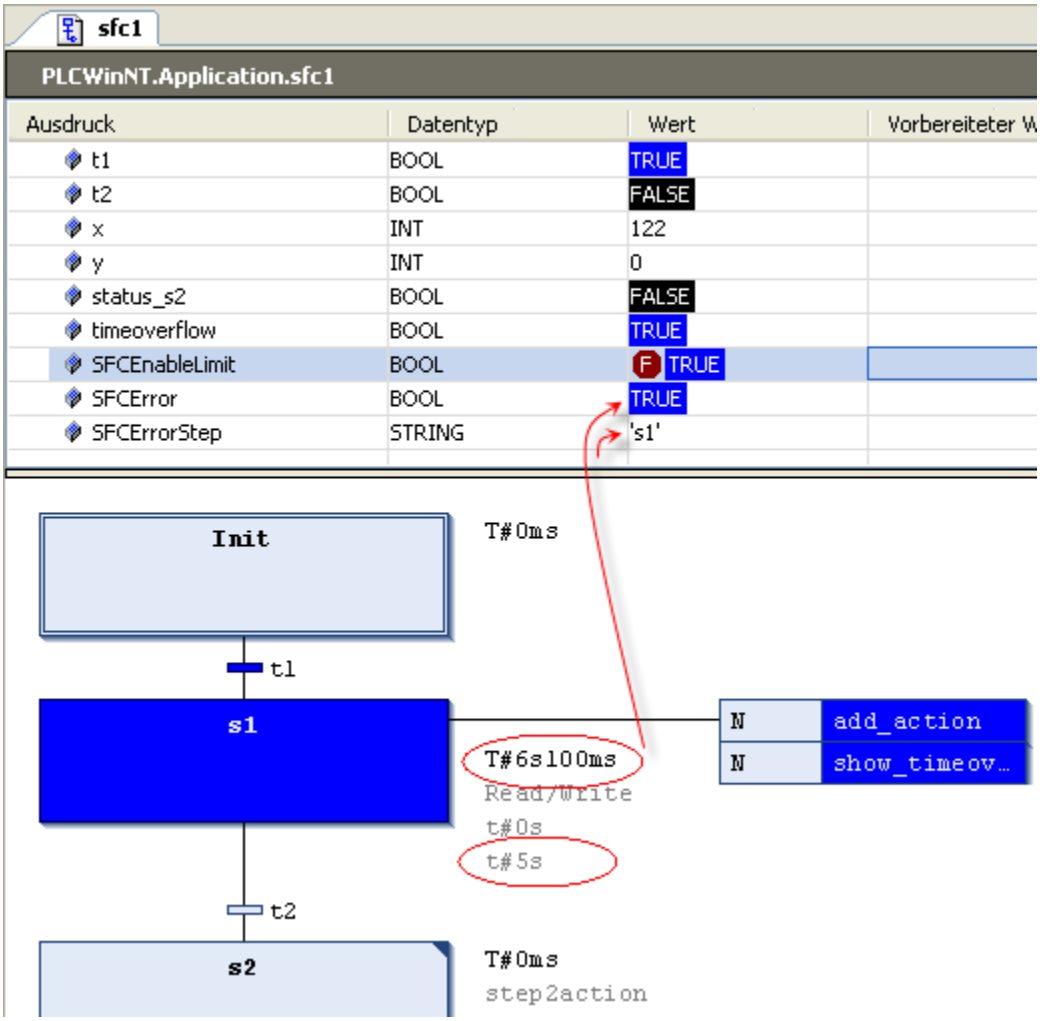


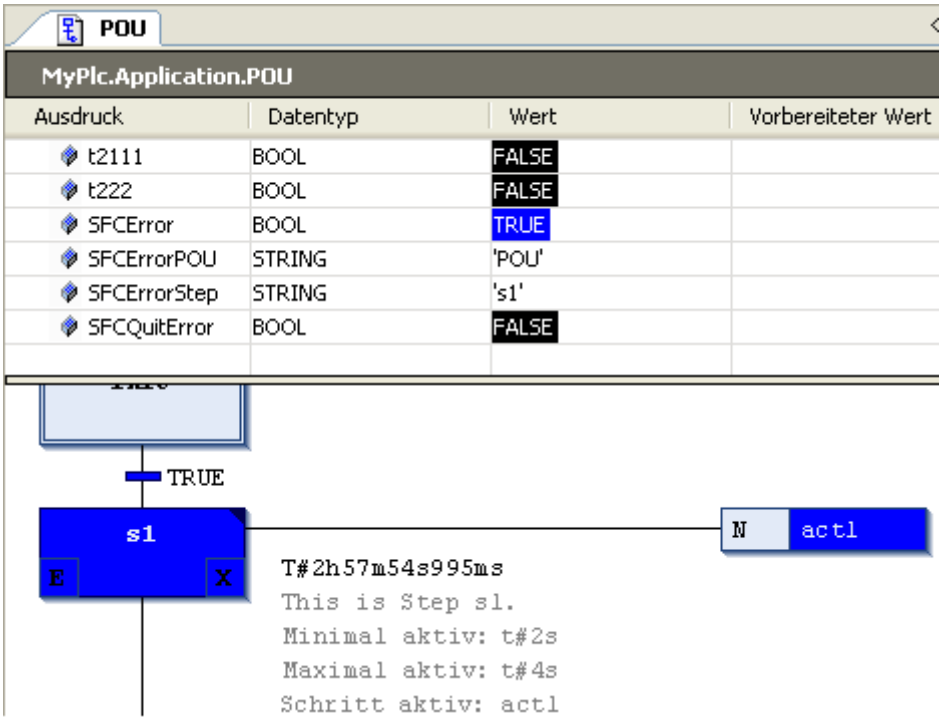
Abb. 405 Online Ansicht von AS.Baustein sfc1

Die folgenden Flag-Variablen können verwendet werden:

SFCInit: BOOL;	Wenn diese Variable TRUE ist, dann wird im Ablauf auf den Initialschritt zurückgesetzt. Die anderen AS-Flags werden ebenfalls zurückgesetzt (Initialisierung). Solange die Variable TRUE ist, bleibt der Initialschritt gesetzt (aktiv), wird aber nicht ausgeführt. Erst wenn SFCInit wieder auf FALSE gesetzt wird, wird der Baustein normal weiter bearbeitet.
SFCReset: BOOL;	Diese Variable verhält sich ähnlich wie SFCInit. Im Unterschied zu dieser wird allerdings nach der Initialisierung der Initialschritt weiter abgearbeitet. So könnte beispielsweise im Init-Schritt das SFCReset-Flag gleich wieder auf FALSE gesetzt werden.
SFCError: BOOL;	Diese Variable wird TRUE, wenn in einem AS-Diagramm eine Zeitüberschreitung aufgetreten ist. Wenn im Programm nach der ersten Zeitüberschreitung eine weitere auftritt, wird diese nicht mehr registriert, wenn die Variable SFCError vorher nicht wieder zurückgesetzt wurde. Die Deklaration von SFCError ist Voraussetzung für das Funktionieren der anderen Flag-Variablen zur Kontrolle des zeitlichen Ablaufs (SFCErrorStep, SFCErrorPOU, SFCQuitError).
SFCEnableLimit: BOOL;	Diese Variable dient dem gezielten Einschalten (TRUE) und Ausschalten (FALSE) der Kontrolle für Zeitüberschreitungen in Schritten durch SFCError. Das heißt, wenn diese Variable deklariert und aktiviert ist (AS-Einstellungen), muss sie auch auf TRUE gesetzt werden, damit SFCError arbeitet, ansonsten werden Zeitüberschreitungen ignoriert. Die Verwendung kann beispielsweise bei Inbetriebnahme oder Handbetrieb nützlich sein. Wenn die Variable nicht deklariert ist, arbeitet SFCError automatisch. Voraussetzung ist natürlich die Deklaration von SFCError.
SFCErrorStep: STRING;	Diese Variable speichert den Namen des Schritts, der eine durch SFCError registrierte Zeitüberschreitung verursacht hat. Voraussetzung ist die Deklaration von SFCError.
SFCErrorPOU: STRING;	Diese Variable speichert im Falle einer Zeitüberschreitung den Namen des Bausteins, in dem eine durch SFCError registrierte Zeitüberschreitung aufgetreten ist. Voraussetzung ist die Deklaration von SFCError.
SFCQuitError: BOOL;	Solange diese boolsche Variable TRUE ist, wird die Abarbeitung des AS-Diagramms angehalten, eine eventuelle Zeitüberschreitung in der Variablen SFCError wird dabei zurückgesetzt. Wenn die Variable wieder auf FALSE gesetzt wird, werden alle bisherigen Zeiten in den aktiven Schritten zurückgesetzt. Voraussetzung dafür ist die Deklaration des Flags SFCError, das die Zeitüberschreitung registriert.
SFCPause: BOOL;	Solange diese Variable TRUE ist, wird die Abarbeitung des AS-Diagramms angehalten.
SFCTrans: BOOL;	Diese Variable wird TRUE, wenn eine Transition schaltet.
SFCCurrentStep: STRING;	Diese Variable zeigt den Name des gerade aktiven Schritts, unabhängig von der Zeitüberwachung. Bei einer Parallel-Verzweigung wird immer der Name des Schrittes des äußersten rechten Zweigs gespeichert.
SFCTip, **SFCTipMode**: BOOL;	Diese Variablen erlauben den "Tip-Betrieb" des AS-Bausteins. Wenn er durch SFCTipMode=TRUE aktiviert ist, kann nur zum nächsten Schritt weiter geschaltet werden, indem SFCTip auf TRUE gesetzt wird. Solange SFCTipMode auf FALSE gesetzt ist, kann zusätzlich auch über die Transitionen weiter geschaltet werden.

Beispiel einiger AS-Flags bei Zeitüberschreitung im Online-Modus des Editors

Die Flag-Variable SFCError registriert eine Zeitüberschreitung in Schritt 's1' im AS-Objekt 'POU':



Zugriff auf die Flags:

Um Zugriff auf die Flags zur Kontrolle der AS-Abarbeitung (Zeitüberschreitungen, Reset, Tip-Modus) zu ermöglichen, müssen die Flag-Variablen wie oben (Kontrolle der Ausführung eines AS-Diagramms) beschrieben, deklariert und aktiviert werden.

Syntax für den Zugriff:

... von einer Aktion oder Transition innerhalb des AS-Bausteins aus:
<Schrittname>.<Flag> resp. **_<Aktionsname>.<Flag>**
Beispiele: "status:=step1._x;" "checkerror:=SFCerror;"

... von einem anderen Baustein aus: **<SFC POU>.<Schrittname>.<Flag>** resp. **<SFC POU>_<Aktionsname>.<Flag>**
Beispiele: "status:=SFC_prog.step1._x;" checkerror:=SFC_prog.SFCerror;"

Zu beachten: Im Falle eines **Schreibzugriffs** von einem anderen Baustein aus muss die implizite Variable zusätzlich explizit als eine **VAR_INPUT** Variable im AS-Baustein oder **global**, beispielsweise in einer GVL, deklariert werden.

Beispiel:

Lokale Deklaration:

```
PROGRAM SFC_prog
VAR_INPUT
  SFCinit:BOOL;
END_VAR
```

oder Globale Deklaration in einer globalen Variablenliste:

```
VAR_GLOBAL
  SFCinit:BOOL;
END_VAR
```

Zugriff auf das Flag in PLC_PRG:

```
PROGRAM PLC_PRG
VAR
  setinit: BOOL;
END_VAR
SFC_prog.SFCinit:=setinit;  //Schreibzugriff auf SFCinit in SFC_prog
```

13.13.8 Abarbeitungsreihenfolge in AS

Zur Laufzeit werden die verschiedenen Aktionstypen in definierter Abfolge ausgeführt, siehe untenstehende Tabelle.

Folgende Begriffe werden verwendet:

- **Aktiver Schritt**: Ein aktiver Schritt ist ein Schritt, dessen Aktion(en) gerade ausgeführt wird/werden. Im Online-Modus werden die Boxen aktiver Schritte blau gefüllt dargestellt.
- **Initialschritt**: Im ersten Zyklus nach Aufruf einer AS-POU wird der Initialschritt automatisch aktiv und die Schrittaktion wird ausgeführt.
- **IEC-Aktionen** werden mindestens zweimal ausgeführt. Das erste Mal nach Aktivwerden des Schrittes, das zweite Mal - aber erst im nächsten Zyklus - nach Deaktiveren des Schrittes.
- **Alternative Verzweigungen**: Wenn der Schritt vor der Verzweigung (vor der horizontalen Linie) aktiv ist, wird jeweils die erste Transition jedes alternativen Zweiges von links nach rechts ausgewertet. Im ersten Zweig, in dem eine Transition gefunden wird, die TRUE liefert, wird der darauf folgende Schritt aktiviert.

- **Parallele Verzweigungen**: Wenn der Schritt vor der Verzweigung (vor der horizontalen Doppellinie) aktiv ist und die Transition vor der Verzweigung TRUE liefert, werden die ersten Schritte in allen Zweigen aktiviert. Die Zweige werden dann gleichzeitig abgearbeitet. Der Schritt nach dem Ende der Verzweigung (nach der horizontalen Doppellinie) wird aktiviert, wenn alle letzten Schritte in den Zweigen aktiv sind und die Transition nach der Doppellinie TRUE liefert.

Die einzelnen Stufen der Abarbeitung entsprechen denen bei »PLC Designer« V2.3:

1. Reset der IEC-Aktionen	Die Kontroll-Flags der IEC-Aktionen werden **zurückgesetzt** (allerdings nicht die Flags von Aktionen, die innerhalb von Aktionen aufgerufen werden!).
2. Ausgangsaktionen ("Schritt deaktiviert") ausführen	Alle Schritte werden in der Reihenfolge, in der sie im AS-Diagramm angeordnet sind (von oben nach unten und von links nach rechts) daraufhin überprüft, ob sie die Bedingung für die Ausführung der "Schritt deaktiviert"-Aktion (Ausgangsaktion) erfüllen, und - wenn dies der Fall ist - werden diese ausgeführt. Eine Ausgangsaktion wird ausgeführt, wenn der Schritt deaktiviert wird, d.h. wenn seine Eingangs- und Schrittaktionen (falls vorhanden) im vorangegangenen Zyklus ausgeführt worden sind und die Bedingung für den nachfolgenden Schritt TRUE liefert.
3. Eingangsaktionen ("Schritt aktiviert") ausführen	Alle Schritte werden in der Reihenfolge, in der sie im AS-Diagramm angeordnet sind (von oben nach unten und von links nach rechts) daraufhin überprüft, ob sie die Bedingung für die Ausführung der "Schritt aktiviert"-Aktion (Eingangsaktion) erfüllen, und - wenn dies der Fall ist - werden diese ausgeführt. Eine Eingangsaktion wird ausgeführt, wenn die Abarbeitung bei der dem Schritt vorangehenden Transition angelangt ist und diese TRUE liefert, der Schritt also aktiviert wird.
4. Zeitenprüfung, Schrittaktionen ("Schritt aktiv") ausführen	Für alle Schritte wird in der Reihenfolge, in der sie im AS-Diagramm angeordnet sind (von oben nach unten und von links nach rechts) folgendes geprüft: - Die verstrichene Zeit des aktiven Schritts wird in die entsprechende implizite Schrittvariable <Schrittname>.t kopiert. - Wenn eine Zeitüberschreitung festgestellt wird, werden die entsprechenden Fehler-Flags bedient. - Bei Nicht-IEC-Schritten wird die **Schrittaktion ("Schritt aktiv")** nun ausgeführt.
5. IEC-Aktionen ausführen	Die IEC-Aktionen werden in alphabetischer Reihenfolge ausgeführt. Dies erfolgt in zwei Durchläufen durch die Liste der Aktionen. Im ersten Lauf werden die IEC-Aktionen aller Schritte ausgeführt, die im vorangegangenen Zyklus deaktiviert wurden. Im zweiten Lauf werden die IEC-Aktionen aller Schritte ausgeführt, die aktiv sind.
6. Transitionen-Check, Aktivierung der nachfolgenden Schritte	Die Transitionen werden ausgewertet: Wenn ein Schritt im aktuellen Zyklus aktiv ist und die nachfolgende Transition TRUE liefert (und die ggfs. definierte Minimalzeit des Schrittes abgelaufen ist), wird der nachfolgende Schritt **aktiviert**.

Achtung:
Folgendes muss bei der Implementierung von Aktionen beachtet werden:
Es kann vorkommen, dass eine Aktion innerhalb desselben Zyklus mehrfach ausgeführt wird, weil sie in mehreren AS-Diagrammen verwendet wird. Beispiel: ein AS enthält zwei IEC-Aktionen A und B, die beide in der Sprache AS programmiert sind und die beide eine IEC-Aktion C aufrufen. In diesem Fall würde C zweimal aufgerufen werden.
Wenn die gleiche IEC-Aktion gleichzeitig auf verschiedenen Ebenen eines AS-Diagramms verwendet wird, kann das zu unvorhersagbaren Effekten während der Abarbeitung führen. Deshalb wird eine entsprechende Fehlermeldung ausgegeben. Diese wird möglicherweise erscheinen, wenn mit Projekten gearbeitet wird, die mit einer älteren Version des Programmiersystems erstellt wurden.

Hinweis: Beachten Sie die Möglichkeit, Implizite Variablen zu verwenden, um den Abarbeitungsstatus von Schritten und Aktionen zu überwachen bzw. die Abarbeitung zu kontrollieren.

13.13.9 AS-Editor im Online-Modus

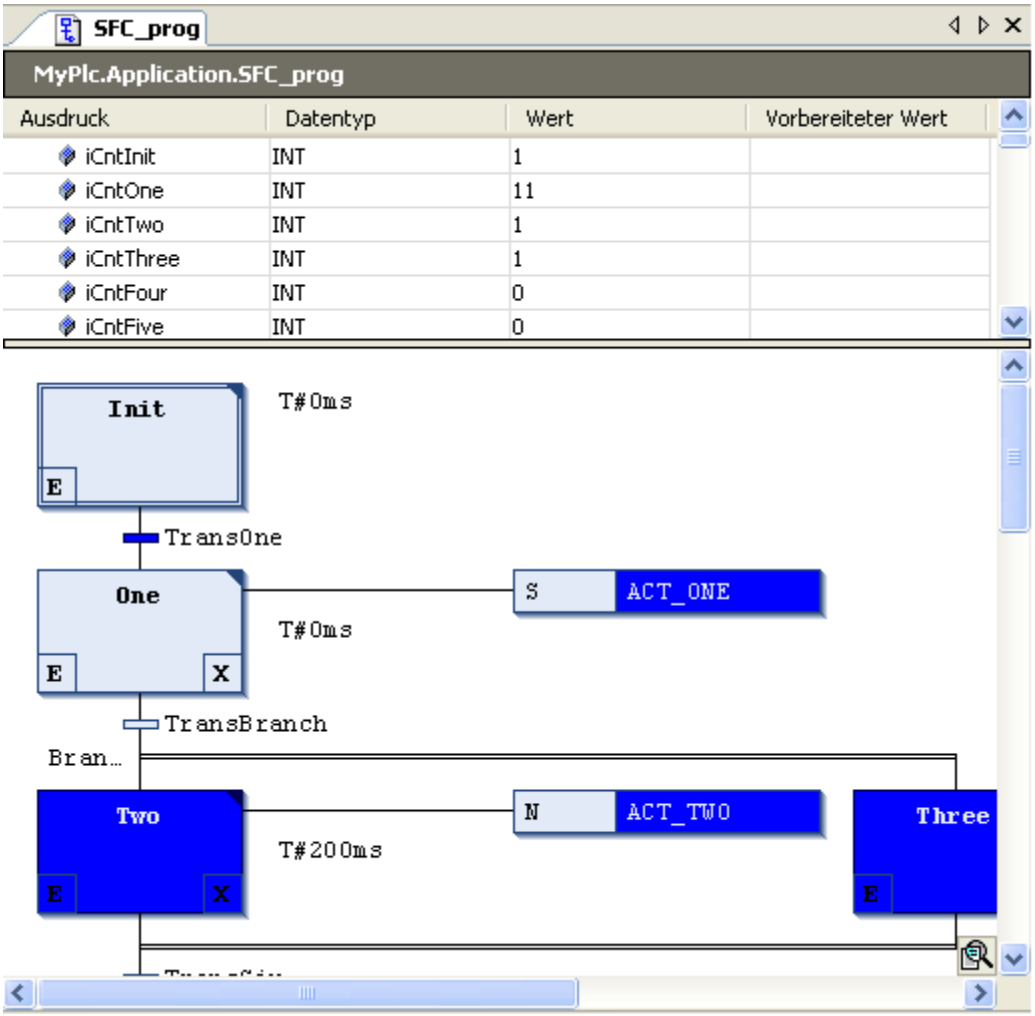
Zur Laufzeit ermöglicht der SFC-Editor die Anzeige (Monitoring) und das Schreiben und Forcen der im AS-Diagramm verwendeten Variablen und Ausdrücke auf der Steuerung. Debugging- Funktionalität (Haltepunkte, Schrittweise Ausführung etc.) ist noch nicht verfügbar. Siehe unten.

- Beachten Sie, dass das Editorfenster eines SFC-Objekts auch den Deklarationseditor im oberen Teil enthält. Falls mittels des AS-Einstellungen-Dialogs implizite Variablen (SFC flags) deklariert wurden, so werden diese hier hinzugefügt, können im Offline Modus im Deklarationsteil jedoch nicht gesehen werden.
- Bitte beachten Sie auch die Abarbeitungsreihenfolge für die Elemente in einem AS-Diagram.
- Einstellungen bezüglich Kompilation und Online-Darstellung der AS-Elemente und Attribute können in den Dialogen der Objekt-Eigenschaften bzw. AS-Editor Optionen und AS-Objekt Optionen vorgenommen werden.
- Der Abarbeitungsprozess eines AS-Diagramms kann mit Hilfe von impliziten Variablen (AS-Flags) überwacht und kontrolliert werden.

Monitoring

Aktive Schritte werden blau gefüllt dargestellt. Die Anzeige der Schrittattribute im AS-Diagramm hängt von den aktuellen Eintstellungen in den SFC-Editor Optionen ab.

Beispiel: Online-Ansicht eines Programms "SFC_prog"



Haltepunkt-Positionen im SFC-Editor: (noch nicht verfügbar)

13.14 ST-Editor

Der ST-Editor, der mit dem Plug-In 'ST Editor' installiert wird, dient dem Programmieren von Objekten in der IEC-Programmiersprache Strukturierter Text (ST) bzw. Erweiterter Strukturierter Text , der bezüglich der Norm IEC 61131-3 einige zusätzliche Funktionen bietet.

Der ST-Editor ist ein **Texteditor** und somit können die entsprechenden Optionen und Anpassungen-Dialoge verwendet werden, um Verhalten, Aussehen und Menüs zu konfigurieren. Farben, Zeilennumerierung, Tabulatorbreiten, Einrückungen etc. können dort eingestellt werden.

Beachten Sie, dass **Block-Selektion** in Texten möglich ist, indem die <Alt>-Taste gedrückt wird, während der gewünschte Textbereich mit der Maus ausgewählt wird.

Der ST-Editor wird im unteren Bereich des Editorfensters geöffnet, das im oberen Teil auch den Deklarationseditor enthält.

Beachten Sie, dass im Fall von **syntaktischen Fehlern** während des Editierens entsprechende Meldungen im "Precompile" Meldungsfenster ausgegeben werden. Dieses Fenster wird immer aktualisiert, wenn das Editorfenster erneut den Eingabefokus erhält (also z.B. wenn der Cursor in ein anderes Fenster und dann zurück in den ST-Editor gesetzt wird).

13.14.1 ST-Editor im Online-Modus

Im Online-Modus stellt der Editor für Strukturierten Text (ST-Editor) Ansichten für das "Monitoring", also Anzeigen, und für das Schreiben und Forcen von Werten auf Variablen und Ausdrücke auf der Steuerung bereit. Debugging-Funktionen (Haltepunkte, Einzelschrittabarbeitung etc.) sind verfügbar. Siehe unten.

Monitoring

Wenn die "**Inline-Monitoring**"-Funktion nicht explizit deaktiviert ist (Optionen-Dialog), wird der aktuelle Variablenwert in einem kleinen Fenster hinter der Variable angezeigt.

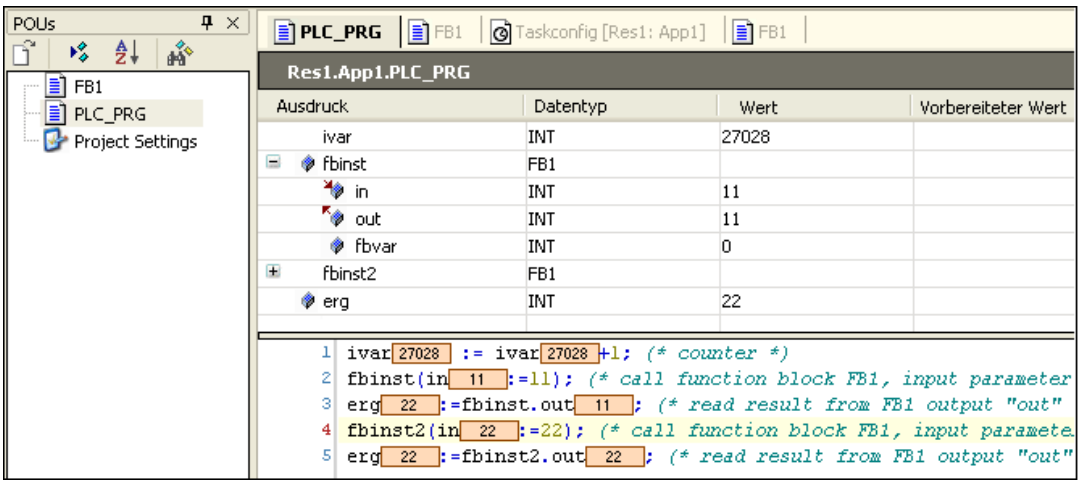


Abb. 406 Beispiel: Online-Ansicht eines Programms "PLC_PRG" mit Inline-Monitoring

Online-Ansicht eines Funktionsblocks:

Nur in der Ansicht einer Instanz ist Monitoring möglich, in der Ansicht der Basis-Implementation werden keine Werte angezeigt. Anstatt enthält die Spalte 'Wert' den Text "<Der Wert des Ausdrucks kann nicht gelesen werden>" und in den Inline-Monitoring-Feldern im Implementationsteil stehen jeweils drei Fragezeichen.

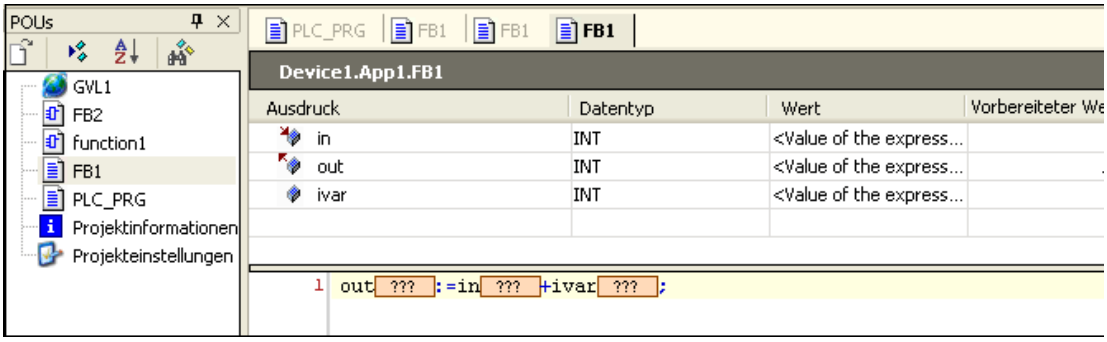


Abb. 407 Beispiel: Online-Ansicht eines Funktionsblocks FB1

Forcen von Variablen

Zusätzlich zu der in jedem Editor möglichen Eingabe eines vorbereiteten Werts für eine Variable im Deklarationsteil, besteht im ST-Editor im Online-Modus die Möglichkeit auch im Implementierungsteil auf die Monitoringbox einer Variablen zu klicken, woraufhin ein Dialog geöffnet wird, in welchem der vorbereitete Wert eingegeben werden kann:

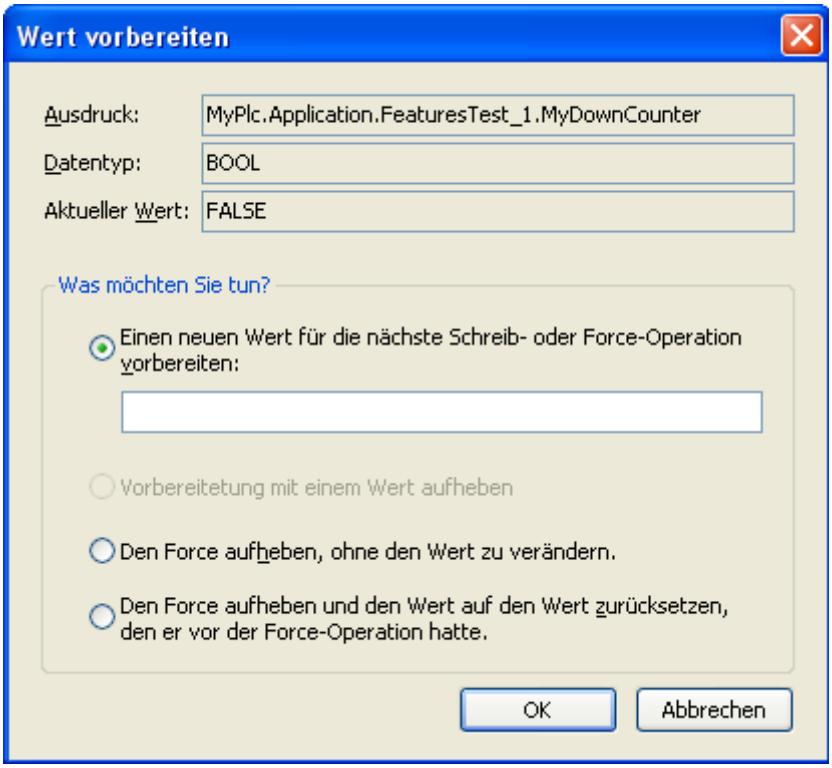


Abb. 408 Dialog - Wert vorbereiten

Von der Variablen wird ihr vollständiger Pfad innerhalb des Gerätebaums, angezeigt, ihr Typ und ihr aktueller Wert. Durch Aktivierung des entsprechenden Punkts können Sie entscheiden, ob

- ein neuer Wert vorbereitet werden soll, der dazu im Editierfeld eingetragen werden muss
- ein vorbereiteter Wert gelöscht werden soll
- eine aktuell geforcte Variable freigegeben werden soll
- eine aktuell geforcte Variable freigegeben und auf ihren ursprünglichen Wert vor dem Forcen gesetzt werden soll

Die ausgewählte Aktion wird nach Aufruf des Kommandos **Werte forcen** (Menüpunkt Online) oder nach Drücken der Taste **F7** ausgeführt.

Haltepunkt-Positionen im ST-Editor:

Der Anwender kann einen Haltepunkt (Breakpoint) grundsätzlich an den Positionen eines Bausteins setzen, an denen sich ein Variablenwert ändern kann oder wo der Programmverlauf sich verzweigt oder ein anderer Baustein aufgerufen wird. In den folgenden Beschreibungen zeigt "{BP}" eine mögliche Haltepunkt-Position an.:

Zuweisung: Am Beginn einer Zeile. Beachten Sie: Bei der Verwendung von Zuweisungen als Ausdrücke entstehen keine weiteren Haltepunkt-Positionen innerhalb einer Zeile.

FOR-Schleife: 1. vor der Initialisierung des Zählers, 2. vor der Überprüfung des Zählers. 3. vor einer Anweisung

{BP} FOR i := 12 TO {BP} x {BP} BY 1 DO

{BP} [statement1]

...

{BP} [statementn-2]

END_FOR

WHILE-Schleife: 1. vor dem Prüfen der Bedingung. 2. vor einer Anweisung

{BP} WHILE i < 12 DO

{BP} [statement1]

...

{BP} [statementn-1]

END_WHILE

REPEAT-Schleife: vor dem Prüfen der Bedingung.

REPEAT

{BP} [statement1]

...

{BP} [statementn-1]

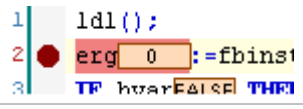
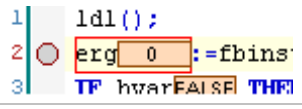
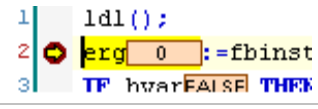
{BP} UNTIL i >= 12

END_REPEAT

Aufruf eines Programms oder Funktionsblocks: Am Beginn der Zeile.

Es gibt immer eine Haltepunkt-Position am Ende eines Bausteins. Beim schrittweisen Abarbeiten (Steppen) wird diese Position nach einer RETURN-Anweisung erreicht.

Haltepunkt-Darstellung in ST:

Haltepunkt im Online-Modus:	Deaktivierter Haltepunkt:	Programm-Stop an einem Haltepunkt:
		

Hinweis: Zu beachten für Haltepunkte in Methoden: Ein Haltepunkt wird automatisch in allen Methoden gesetzt, die aufgerufen werden können. Somit gilt: Wenn eine Methode aufgerufen wird, die über eine Schnittstelle definiert ist, werden in allen Methoden von Funktionsblöcken, die diese Schnittstelle implementieren, Haltepunkte gesetzt und ebenfalls in allen abgeleiteten Funktionsblöcken, die die Methode definieren.

13.14.2 Strukturierter Text ST / ExST

Strukturierter Text ST / Erweiterter Strukturierter Text ExST

Der Strukturierte Text ist eine Programmiersprache, vergleichbar mit anderen Hochsprachen wie C oder PASCAL, die die Entwicklung komplexer Algorithmen erlaubt. Der Progamm-Code besteht aus einer Kombination von Ausdrücken und Anweisungen, die bedingt ("IF..THEN..ELSE) oder in Schleifen (WHILE..DO) ausgeführt werden können.

Beispiel:

```
IF value < 7 THEN
 WHILE value < 8 DO
  value:=value+1;
 END_WHILE;
END_IF;
```

Erweiterter Strukturierter Text (ExST) ist eine »PLC Designer«-spezifische Erweiterung bezüglich des IEC 61131-3 Standards für Strukturierten Text (ST). Beispiele: Zuweisung als Ausdruck, Set-/Reset-Operatoren.

Ausdrücke

Ein "Ausdruck" ist ein Konstrukt, das nach seiner Auswertung einen Wert zurück liefert.

Ausdrücke sind zusammengesetzt aus Operatoren und Operanden. Auch Zuweisungen können als Ausdruck verwendet werden. Ein Operand kann eine Konstante, eine Variable, ein Funktionsaufruf oder ein weiterer Ausdruck sein.

Beispiele:

33	(* Konstante *)
ivar	(* Variable *)
fct(a,b,c)	(* Funktionsaufruf *)
a AND b	(* Ausdruck *)
(x*y) / z	(* Ausdruck *)
real_var2 := int_var;	(* Zuweisung, s.u. *)

Auswertung von Ausdrücken

Die Auswertung eines Ausdrucks erfolgt durch Abarbeitung der Operatoren nach bestimmten Bindungsregeln. Der Operator mit der stärksten Bindung wird zuerst abgearbeitet, dann der Operator mit der nächststärkeren Bindung, usw., bis alle Operatoren abgearbeitet sind.

Operatoren mit gleicher Bindungsstärke werden von links nach rechts abgearbeitet.

Nachfolgend finden Sie eine Tabelle der ST-Operatoren in der Ordnung ihrer Bindungsstärke:

Operation	Symbol	Bindungsstärke
Einklammern	(Ausdruck)	Stärkste Bindung
Funktionsaufruf	Funktionsname (Parameterliste)	
Potenzieren	EXPT	
Negieren Komplementbildung	- NOT	
Multiplizieren Dividieren Modulo	* / MOD	
Addieren Subtrahieren	+ -	
Vergleiche	<,>,<=,>=	
Gleichheit Ungleichheit	= <>	
Bool AND	AND	
Bool XOR	XOR	
Bool OR	OR	Schwächste Bindung

Zuweisung als Ausdruck

Als Erweiterung bezüglich der Norm IEC 61131-3(ExST) erlaubt »PLC Designer« die Verwendung von Zuweisungen als Ausdrücke.

Beispiele:

int_var1 := int_var2 := int_var3 + 9;	(* int_var1
real_var1 := real_var2 := int_var;	(* korrekte Zuweisungen, real_var1 und real_var2 erhalten den Wert von int_var *)
int_var := real_var1 := int_var;	(* fehlerhafte Zuweisung, die Datentypen stimmen nicht überein ! *)

IF b := (i = 1) THEN
i := i + 1;
END_IF	

Anweisungen

Anweisungen steuern, wie die gegebenen Ausdrücke verarbeitet werden sollen. Folgende Anweisungen können in ST verwendet werden:

Instruction	Beispiel
Zuweisung (siehe auch: Zuweisungsoperatoren)	A:=B; CV := CV + 1; C:=SIN(X);
Aufruf eines Funktionsblocks und Verwendung des FB-Ausgangs	CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
RETURN	RETURN;

IF	D:=B*B;
	IF D<0.0 THEN
	C:=A;
	ELSIF D=0.0 THEN
	C:=B;
	ELSE
	C:=D;
	END_IF;

CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;
FOR	J:=101;

Instruction	Beispiel

	FOR I:=1 TO 100 BY 2 DO
	IF ARR[I] = 70 THEN
	J:=I;
	EXIT;
	END_IF;
	END_FOR;

WHILE	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT;
CONTINUE	CONTINUE;
JMP	label: i:=i+1; JMP label;
Leere Anweisung	;

Zuweisungsoperatoren

Auf der linken Seite einer Zuweisung steht ein Operand (Variable, Adresse) der der Wert des Ausdrucks auf der rechten Seite mit Hilfe eines Zuweisungsoperators **:=** zugewiesen wird.

Siehe auch: MOVE-Operator, der dasselbe bewirkt.

Beispiel:

```
Var1 := Var2 * 10;
```

Nach Ausführung dieser Zeile hat Var1 den zehnfachen Wert von Var2.

Beachten Sie die folgenden Funktionalitäten, die neu sind im Vergleich zu »PLC Designer« V2.3:

Weitere Zuweisungsoperatoren, die nicht in der Norm 61131-3 beschrieben sind (**ExST**):

Set operator '**S=**': Der Wert wird "gesetzt", d.h. wenn er einmal TRUE wurde,bleibt er TRUE.

Beispiel:

```
a S= b;
```

a erhält den Wert von b; wenn a einmal auf TRUE gesetzt wurde, bleibt es TRUE, auch wenn b wieder FALSE wird.

Reset operator '**R=**': Der Wert wird "zurückgesetzt", d.h. wenn er einmal FALSE wurde,bleibt er FALSE.

Beispiel:

```
a R= b;
```

a wird auf FALSE gesetzt, sobald b = TRUE.

Hinweis: Beachten Sie das Verhalten im Falle von Mehrfach-Zuweisungen. Alle "Set"- und "Reset"-Zuweisungen beziehen sich immer auf das letzte Glied der Zuweisung. Beispiel: a S= b R= fun1(par1,par2);

In diesem Fall erhält b die durch "Reset" resultierende Ausgabe von fun1, ABER: a erhält nicht das "Set"-Ergebnis aus b, sondern das "Set"-Ergebnis aus fun1!

Zuweisung als Ausdruck, Erweiterung zur Norm IEC 61131-3 (ExST**):**

Beachten Sie, dass eine Zuweisung als Ausdruck verwendet werden kann.

Aufruf eines Funktionsblocks in ST

Ein Funktionsblock (im folgenden auch mit "FB" abgekürzt) wird in ST gemäß folgender Syntax aufgerufen:

Syntax:

<FB-instanz>(FB Eingabevariable:=<Wert oder Adresse>|, <weitere FB Eingabevaribale:=<Wert oder Adresse>|...weitere FB Eingabevariablen);

Beispiel:

Im folgenden Beispiel wird ein Timer-FB (TON) mit Zuweisungen für die Parameter IN und PT aufgerufen.

Die Ausgabevariable Q wird danach Variable A zugewiesen. Der Timer-FB wird in "TMR:TON;" instanziert.

Die Ausgabevariable wird gemäß Syntax <FB-Instanz>.<FB-Variable> angesprochen.

```
TMR(IN := %IX5, PT := 300);
A:=TMR.Q
```

RETURN-Anweisung

Die RETURN-Anweisung kann verwendet werden, um einen Baustein zu verlassen, z.B. abhängig von einer Bedingung.

Syntax:

RETURN;

Beispiel:

```
IF b=TRUE THEN
RETURN;
END_IF;
a:=a+1;
```

Wenn b den Wert TRUE hat, wird Anweisung a:=a+1; nicht ausgeführt, sondern der Baustein wird sofort verlassen.

IF-Anweisung

Mit der IF-Anweisung kann eine Bedingung geprüft werden und, abhängig von dieser Bedingung, Anweisungen ausgeführt werden.

Syntax:

IF <Boolscher_Ausduck_1> **THEN**

<IF_Anweisungen>

{ELSIF <Boolscher_Ausduck_2> **THEN**

<ELSIF_Anweisung_1>

.

ELSIF <Boolscher_Ausduck_n> **THEN**

<ELSIF_Anweisung_n-1>

ELSE

<ELSE_Anweisungen>**}**

END_IF;

Der Teil in geschweiften Klammern {} ist optional.

Wenn <Boolean_expression1> TRUE liefert, dann werden nur die <IF_Instructions> und keine der anderen Anweisungen ausgeführt.

Ansonsten werden die boolschen Ausdrücke, beginnend mit <Boolean_expression2>, nacheinander geprüft bis ein Ausdruck TRUE liefert. Dann werden alle Anweisungen, die zwischen diesem Ausdruck und vor der nächsten ELSE or ELSIF Anweisung stehen ausgewertet und entsprechend ausgeführt.

Wenn keine der boolschen Ausdrücke TRUE liefert, dann werden nur die <ELSE_instructions> ausgewertet.

Beispiel:

```
IF temp<17
THEN heating_on := TRUE;
ELSE heating_on := FALSE;
END_IF;
```

Hier wird die Heizung angeschaltet, wenn die Temperatur unter 17 Grad sinkt. Ansonsten bleibt sie ausgeschaltet.

CASE-Anweisung

Mit einer CASE-Anweisung können mehrere bedingte Anweisungen mit derselben Bedingungsvariablen in einem Konstrukt zusammengefaßt werden.

Syntax:

CASE <Var1> **OF**

<wert1>**:** <anweisung1>

<wert2>**:** <anweisung2>

<wert3, wert4, wert5>: <anweisung3>

<wert6 .. wert10>: <anweisung4>

...

<wert n>**:** <anweisung n>

ELSE <ELSE anweisung>

END_CASE;

Eine CASE-Anweisung wird nach folgendem Schema abgearbeitet:

- Wenn die Variable <Var1> den Wert <wert i> hat, dann wird die Anweisung <Anweisung i> ausgeführt.
- Wenn <Var1> keinen der angegebenen Werte hat, dann wird die <ELSE anweisung> ausgeführt.
- Wenn für mehrere Werte der Variablen, dieselbe Anweisung auszuführen ist, dann kann man diese Werte mit Kommata getrennt hintereinander schreiben, und damit die gemeinsame Anweisung bedingen.
- Wenn für einen ganzen Wertebereich der Variablen dieselbe Anweisung auszuführen ist, dann kann man den Anfangs- und Endwert getrennt durch zwei Punkte hintereinander schreiben, und damit die gemeinsame Anweisung bedingen.

Beispiel:

```
CASE INT1 OF
1, 5: BOOL1 := TRUE;
 BOOL3 := FALSE;
2: BOOL2 := FALSE;
 BOOL3 := TRUE;
10..20: BOOL1 := TRUE;
 BOOL3:= TRUE;
ELSE
 BOOL1 := NOT BOOL1;
 BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

FOR-Schleife

Mit der FOR-Schleife können wiederholte Vorgänge programmiert werden.

Syntax:

INT_Var :INT;

FOR <INT_Var> := <INIT_VALUE> **TO** <END_VALUE> **{BY** <schrittgröße>**} DO**

<anweisungen>

END_FOR;

Der Teil in geschweiften Klammern {} ist optional.

Die <anweisungen> werden solange ausgeführt wie der Zähler <INT_Var> nicht größer als der <END_WERT> ist. Dies wird vor der Ausführung der <anweisungen> überprüft, so dass die <anweisungen> niemals ausgeführt werden, wenn der Anfangswert <INIT_WERT> größer als der Endwert <END_WERT> ist.

Immer wenn <anweisungen> ausgeführt worden ist, wird <INT_Var> um <Schrittgröße> erhöht. Die Schrittgröße kann jeden ganzzahligen Wert haben. Fehlt die Angabe, wird die Schrittgröße auf Standardwert "1" gesetzt. Dies garantiert, dass die Schleife auch zum Ende kommt, da <INT_Var> nur größer werden kann.

Beispiel:

```
FOR Counter:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Nehmen wir an, dass die Variable Var1 mit dem Wert "1" vorbelegt wurde, dann wird sie nach der FOR-Schleife den Wert "32" haben.

Achtung: Wenn <END_VALUE> gleich dem Grenzwert des Zählers <INT_VAR> ist, z.B. wenn Counter - im oben gezeigten Beispiel verwendet - vom Typ SINT ist und wenn <END_VALUE> gleich 127 ist, dann entsteht eine Endlosschleife. **Also darf <END_VALUE> nicht den gleichen Wert erhalten wie der Grenzwert des Zählers!**

Erweiterung bezüglich des IEC 61131-3 Standards (**ExST**):

Die CONTINUE-Anweisung kann innerhalb einer FOR-Schleife verwendet werden.

WHILE-Schleife

Die WHILE-Schleife kann benutzt werden wie die FOR-Schleife, mit dem Unterschied, dass die Abbruchbedingung ein beliebiger boolscher Ausdruck sein kann. Das heißt, man gibt eine Bedingung an, die, wenn sie zutrifft, die Ausführung der Schleife zur Folge hat.

Syntax:

WHILE <boolescher Ausdruck> DO

<Anweisungen>

END_WHILE;

Die <Anweisungen> werden solange wiederholt ausgeführt, wie <boolscher Ausdruck> TRUE ergibt. Wenn <boolscher Ausdruck> bereits bei der ersten Auswertung FALSE ist, dann werden die <Anweisungen> niemals ausgeführt. Wenn <boolscher Ausdruck> niemals den Wert FALSE annimmt, dann werden die <Anweisungen> endlos wiederholt, wodurch ein Laufzeitfehler entsteht.

Hinweis: Der Programmierer muss sicherstellen, dass keine Endlos-Schleife verursacht wird, indem er im Anweisungsteil der Schleife die Bedingung verändert, also zum Beispiel einen Zähler nach oben oder unten zählt.

Beispiel:

```
WHILE Counter<>0 DO
 Var1 := Var1*2;
 Counter := Counter-1;
END_WHILE
```

Die WHILE- und die REPEAT-Schleife sind in gewissem Sinne mächtiger als die FOR-Schleife, da man nicht bereits vor der Ausführung der Schleife die Anzahl der Schleifendurchläufe wissen muss. In manchen Fällen wird man also nur mit diesen beiden Schleifenarten arbeiten können. Wenn jedoch die Anzahl der Schleifendurchläufe klar ist, dann ist eine FOR- Schleife zu bevorzugen, da sie keine endlosen Schleifen ermöglicht.

Erweiterung bezüglich des IEC 61131-3 Standards (**ExST**):

Die CONTINUE-Anweisung kann innerhalb einer WHILE-Schleife verwendet werden.

REPEAT-Schleife

Die REPEAT-Schleife unterscheidet sich von den WHILE-Schleifen dadurch, dass die Abbruchbedingung erst nach dem Ausführen der Schleife überprüft wird. Das hat zur Folge, dass die Schleife mindestens einmal durchlaufen wird, egal wie die Abbruchbedingung lautet.

Syntax:

REPEAT

<Anweisungen>

UNTIL <boolscher Ausdruck>

END_REPEAT;

Die <Anweisungen> werden solange ausgeführt, bis <boolscher Ausdruck> TRUE ergibt.

Wenn <boolscher Ausdruck> bereits bei der ersten Auswertung TRUE ergibt, dann werden <Anweisungen> genau einmal ausgeführt. Wenn <boolscher Ausdruck> niemals den Wert TRUE annimmt, dann werden die <Anweisungen> endlos wiederholt, wodurch ein Laufzeitfehler entsteht.

Hinweis: Der Programmierer muss selbst dafür sorgen, dass keine Endlos-Schleife entsteht, indem er im Anweisungsteil der Schleife die Bedingung verändert, also zum Beispiel einen Zähler nach oben oder unten zählt.

Beispiel:

```
REPEAT
 Var1 := Var1*2;
 Counter := Counter-1;
UNTIL
 Counter=0
END_REPEAT;
```

Erweiterung bezüglich des IEC 61131-3 Standards (**ExST**):

Die CONTINUE-Anweisung kann innerhalb einer WHILE-Schleife verwendet werden.

CONTINUE-Anweisung

Als Erweiterung bezüglich der Norm IEC 61131-3 (**ExST**) wird die CONTINUE-Anweisung innerhalb von FOR-, WHILE- und REPEAT-Schleifen unterstützt.

CONTINUE bewirkt, dass die Ausführung zum Beginn des nächsten Schleifendurchlaufs springt.

Beispiel:

```
FOR Counter:=1 TO 5 BY 1 DO
INT1:=INT1/2;
IF INT1=0 THEN
CONTINUE; (* um eine Division durch Null zu verhindern *)
END_IF
Var1:=Var1/INT1; (* wird nur ausgeführt, wenn INT1 nicht "0" ist *)
END_FOR;
Erg:=Var1;
```

EXIT-Anweisung

Wenn die EXIT-Anweisung in einer FOR-, WHILE- oder REPEAT-Schleife vorkommt, dann wird die innerste Schleife beendet, ungeachtet der Abbruchbedingung.

JMP-Anweisung

Die JMP-Anweisung kann für einen unbedingten Sprung an eine durch eine Sprungmarke ausgezeichnete Programmzeile benutzt werden.

Syntax:

<Sprungmarke>:

JMP <Sprungmarke>;

Die <**Sprungmarke**> ist ein beliebiger, jedoch eindeutiger Bezeichner, der an den Anfang einer Programmzeile gestellt wird. Der Sprunganweisung JMP folgt die Angabe eines Sprungziels, welches mit dem Namen einer definierten Sprungmarke übereinstimmen muss. Bei Erreichen der JMP Anweisung erfolgt ein Rücksprung an die mit der zum Sprungziel passenden Marke ausgestatteten Programmzeile.

Hinweis: Der Programmierer muss sicherstellen, dass keine Endlos-Schleife verursacht wird, indem er den Sprung beispielsweise an eine Bedingung knüpft.

Beispiel:

```
aaa:=0;
_label1: aaa:=aaa+1;
(*instructions*)
IF (aaa < 10) THEN
JMP _label1;
END_IF;
```

Solange die mit 0 initialisierte Variable i einen Wert kleiner 10 hat, bewirkt die bedingte Sprunganweisung in obigem Beispiel einen wiederholten Rücksprung an die mit der Sprungmarke label versehene Programmzeile und somit eine wiederholte Abarbeitung der zwischen Marke und Sprunganweisung enthaltenen Anweisungen. Da diese auch das Inkrementieren der Variablen i enthalten, ist sichergestellt, dass i die Sprungbedingung (genau bei der neunten Abfrage) verletzt und im Programmablauf fortgefahren wird.

Dieselbe Funktionalität kann im Beispiel durch eine WHILE oder REPEAT Schleife erreicht werden. Generell können und sollten Sprunganweisungen vermieden werden, da sie die Lesbarkeit des Codes vermindern.

Kommentare in ST

Es gibt zwei Möglichkeiten, Kommentare im Strukturierten Text einzufügen.

1. Der Kommentar beginnt mit „(*" und schließt mit „*)". Dies ermöglicht das Einfügen von Kommentaren, die über mehrere Zeilen laufen. Beispiel: "(*This is a comment.*)"

2. Einzeiliger Kommentar (Erweiterung bezüglich der Norm 61131-3): Der Kommentar beginnt mit "//" und endet am Ende der Zeile. Beispiel: "// This is a comment."

Kommentare können überall innerhalb des Deklarationseditors oder ST-Editors platziert werden.

Verschachtelte Kommentare: Kommentare können innerhalb von anderen Kommentaren eingefügt werden.

Beispiel:

```
(*
a:=inst.out; (* to be checked *)
b:=b+1;
*)
```

In diesem Beispiel wird der Kommentar, der mit der ersten öffnenden Klammer beginnt, nicht durch die schließdende Klammer hinter "checked" geschlossen, sondern erst durch diejenige in der letzten Zeile.

13.15 Symbolkonfigurationseditor

Mit der Symbolkonfiguration können Symbolbeschreibungen für Projektvariablen erstellt werden, über die diese Variablen (Item oder Symbole) von außerhalb, z. B. über einen OPC-Server, adressiert werden können.

Die Konfiguration von Symbolen für eine Applikation kann im **Symbolkonfigurations-Editor** vorgenommen werden, der sich öffnet, wenn ein Doppelklick auf das Symbolkonfigurationsobjekt im Geräte-Fenster ausgeführt wird.

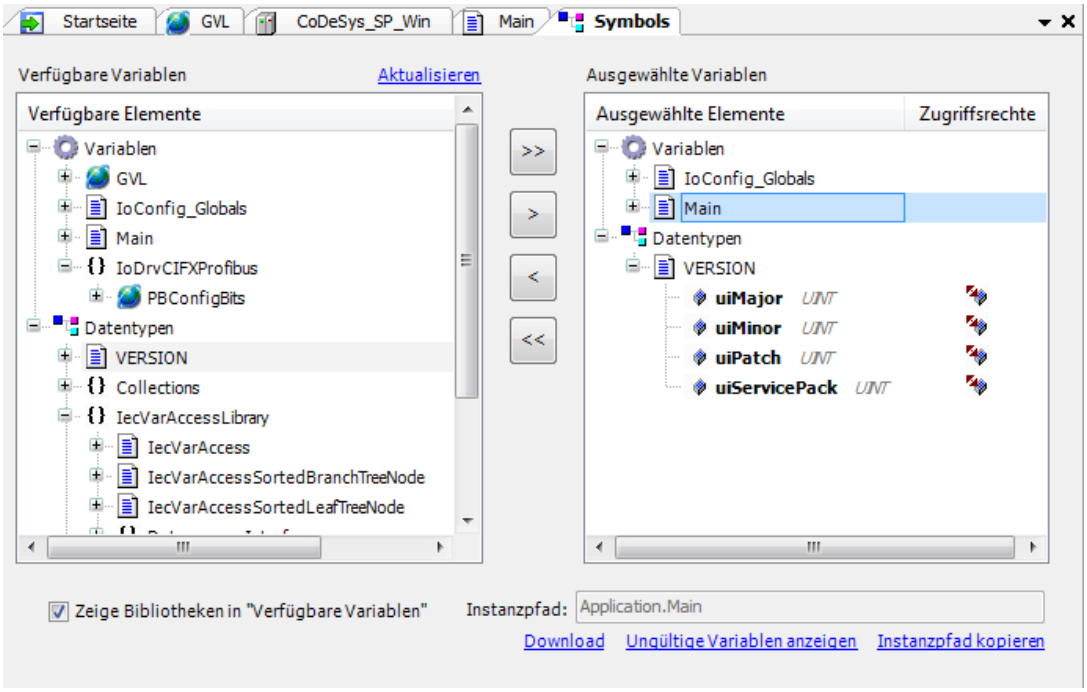


Abb. 409 Symbolkonfiguration, Editor

Im linken Teil des Editor-Fensters werden die gerade verfügbaren Variablen (**Verfügbare Variablen**) der Applikation in einer Baumstruktur dargestellt, die einen Knoten für Variablen (**Variablen**) und einen für Datentypen (**Datentypen**) enthalten. Eingerückt unterhalb dieses Knotens stehen die Namen der Bausteine, globalen Variablenlisten bzw. Datentypen, und wiederum darunter schließlich die Namen, Datentypen und Zugriffsrechte der daraus verfügbaren einzelnen Variablen ("Items"). Mit der Schaltfläche **Aktualisieren** kann der Baum mit den aktuell gültigen Variablen aktualisiert werden. Über die Compiler-Option **Immer binden** kann ein nicht übersetztes Objekt explizit kompiliert und auf die Steuerung geladen werden. Damit sind alle in dem Objekt deklarierten Variablen verfügbar. Außerdem steht das Pragma {attribute 'linkalways') zur Verfügung, das ebenfalls dazu führt, nicht verwendete Variablen in der Symbolkonfiguration zur Verfügung zu stellen.

Im rechten Teil des Editor-Fensters werden die gerade für die Symbolkonfiguration ausgewählten Variablen (**Ausgewählte Variablen**) ebenfalls in einer Baumstruktur dargestellt. Für jedes einzelne Item wird eine Symboldefinition angelegt und in eine Symboldatei exportiert. Wenn die Schaltfläche **Ungültige Variablen anzeigen** betätigt wird, werden alle Variablen rot markiert, die gerade in der Applikation nicht gültig sind, z. B. weil ihre Deklaration gelöscht wurde.

Um eine gerade im linken Baum ausgewählte Variable oder einen Knoten im rechten Baum hinzuzufügen bzw. um einen Eintrag von dort wieder zu entfernen, verwenden Sie die **Einzelpfeil-Schaltflächen** zwischen den beiden Fenstern. Die **Doppelpfeil-Schaltflächen** nehmen jeweils den kompletten Baum, ohne dass vorher etwas ausgewählt werden muss.

Das Zugriffsrecht für ein ausgewähltes Item kann im rechten Fenster verändert werden, indem mit der Maus auf das Symbol in der Spalte **Zugriffsrechte** geklickt wird. Jeder Mausklick schaltet zum nächsten der folgenden möglichen Symbole und dem repräsentierten Recht weiter: read+write (Schreib- und Lesezugriff), write-only (nur Schreibzugriff), read-only (nur Lesezugriff).

Mit der Option **Zeige Bibliotheken in "Verfügbare Variablen"** kann festgelegt werden, ob Variablen von Bibliotheken angezeigt werden.

Der **Instanzpfad** eines gerade ausgewählten Items wird im gleichnamigen Feld im unteren Teil des Dialogs dargestellt. Er kann ausgewählt und mit Hilfe der Schaltfläche **Instanzpfad kopieren** in die Zwischenablage kopiert werden, was hilfreich ist, um einen langen Pfad in ein beliebiges Eingabefeld zu bringen.

Wenn die Symbolkonfiguration im Online-Modus geändert wird, kann die veränderte Applikation mit Hilfe der Schaltfläche **Download** auf die Steuerung geladen werden.

13.16 Taskkonfigurationseditor

Die Task-Konfiguration definiert eine oder mehrere Tasks, die die Ausführung eines Applikations-Programms kontrollieren. Dies ist eine unabdingbare Ressource für eine Applikation und somit MUSS ein Objekt 'Taskkonfiguration' unterhalb einer Applikation im Gerätebaum eingefügt sein..

Im Unterschied zu »PLC Designer« V2.3 wird die Baumstruktur der Taskkonfiguration nicht mit im Editorfenster angezeigt und bearbeitet, sondern ist **in den Gerätebaum im** Geräte**-Fenster integriert**. Sie muss dort unterhalb einer Applikation verfügbar sein und kann über den Befehl Objekt hinzufügen eingefügt werden.

Der oberste Knoten einer Taskkonfiguration heißt immer "Taskkonfiguration". Darunter hängen die definierten Tasks, jede durch ihren Namen repräsentiert. Die Programmaufrufe der einzelnen Tasks werden im Baum nicht angezeigt.

In der Taskkonfguration können die üblichen Befehle für das Bearbeiten eines Objektbaums verwendet werden, um Tasks hinzuzufügen, zu kopieren, auszuschneiden oder zu löschen. Zum Einfügen einer neuen Task beispielsweise wird Befehl Objekt hinzufügen verwendet .

Die einzelnen Tasks werden im **Task Editor** in Dialogen konfiguriert, die auch eine Online-Ansicht bereitstellen. Die Konfigurationsmöglichkeiten sind zielsystem-abhängig.

Abb. 410 Taskkonfiguration unterhalb einer Applikation im Gerätebaum

Eine **Task** ist eine zeitliche Ablaufeinheit eines IEC-Programms. Sie ist definiert durch einen Namen, eine Priorität und einen Typ, der festlegt, welche Bedingung Ihren Start auslöst. Diese Bedingung kann entweder zeitlich definiert sein (Zyklusintervall, freilaufend) oder durch ein internes oder ein externes Ereignis, bei dessen Eintreten die Task ausgeführt werden soll; beispielsweise die steigende Flanke einer globalen Projektvariable oder ein Interrupt-Event der Steuerung.

Jeder Task kann eine Folge von Programmen zugeordnet werden, die beim Ausführen der Task pro Zyklus abgearbeitet werden sollen.

Durch die Kombination von Priorität und Bedingung wird festgelegt, in welcher zeitlichen Abfolge die Tasks ausgeführt werden.

Für jede Task kann eine Zeitüberwachung (Watchdog) konfiguriert werden; die dabei möglichen Einstellungen hängen vom Zielsystem ab.

Zusätzlich gibt es die Möglichkeit, Systemereignisse (z.B. Start, Stop, Reset) direkt mit der Ausführung eines Projektbausteins zu koppeln.

Wenn es vom Zielsystem unterstützt wird, erlaubt die Profiling-Funktionalität eine dynamische Analyse der Ausführungszeit, der Anzahl der Aufrufe und der Code-Abdeckung für die Bausteine, die durch eine Task kontrolliert werden.

Im Online-Modus kann die Task-Abarbeitung in einer speziellen Ansicht überwacht werden.

13.16.1 Task Editor, Verwendung

Das Plug-In 'TaskConfig Editor' stellt den Editor bereit, mit dem die Taskkonfiguration für eine Applikation erstellt werden kann.

Durch einen Doppelklick auf das Objekt Taskkonfiguration im Gerätebaum bzw. durch Befehl 'Objekt bearbeiten' wird das Fenster des Task-Editors geöffnet.

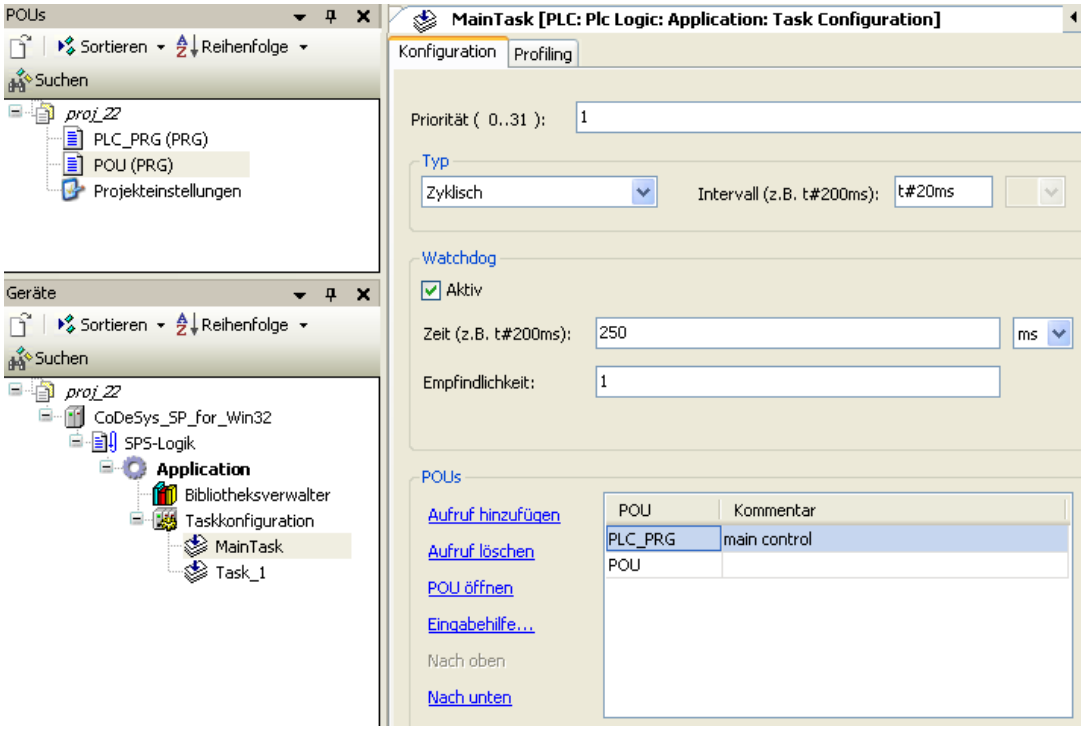


Abb. 411 Task-Editor, Dialog zur Konfiguration einer Task

Im Unterschied zu »PLC Designer« V2.3 ist die Baumstruktur der Taskkonfiguration im Objektbaum im **Geräte**-Fenster integriert. Das übergeordnete Objekt 'Taskkonfiguration' und darunter die zugehörigen Task-Objekte können mit Hilfe des Befehls Objekt hinzufügen eingefügt werden. Wenn ein Taskkonfiguration-Objekt eingefügt wird, wird automatisch eine Task "Task" mit angelegt und der Konfigurationsdialog geöffnet.

Die Taskkonfiguration bzw. eine Task können in den folgenden Dialogen des Task Editors bearbeitet werden. Durch einen Doppelklick auf das entsprechende Objekt im Gerätebaum kann gezielt der jeweilige Dialog geöffnet werden:

- Eigenschaften, zur Anzeige von einigen Grundeinstellungen der Konfiguration.
- Konfiguration, für das Konfigurieren einer einzelnen Task, NOCH NICHT IMPLEMENTIERT: optional erweitert durch den Parameter- Dialog für herstellerspezifische Task-Parameter
- Nur wenn vom Zielsystem unterstützt: Profiling, für das Konfigurieren (Offline-Modus) und Anzeigen (Online-Modus) der dynamischen Analyse einer Applikations-Task während des Online-Betriebs.
- Systemereignisse, für die Verknüpfung von Bausteinaufrufen mit Systemereignissen.

Es hängt vom verwendeten Zielsystem ab, welche Konfigurationsmöglichkeiten in den Dialogen bereitstehen.

Hinweis: Verwenden Sie nicht dieselbe String-Funktion (siehe Standard.library) in mehreren Tasks, da dies zu Programmfehlern durch Überschreiben führen kann.

Dialog 'Eigenschaften'

Wenn der oberste Knoten im Baum der Taskkonfiguration ausgewählt ist, wird der Dialog **Eigenschaften** im Task-Editor-Fenster geöffnet.

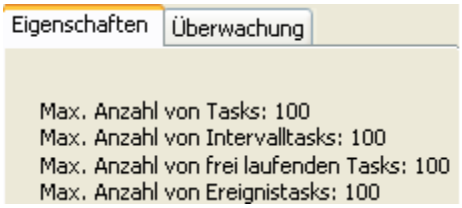


Abb. 412 Taskkonfiguration, Eigenschaften-Dialog, Beispiel

Hier findet man eine Anzeige von allgemeinen Einstellungen zur Taskkonfiguration, die vom Zielsystem vorgegeben werden. Dies ist beispielsweise die maximal mögliche Anzahl von Tasks pro Task-Typ.

Dialog 'Konfiguration'

Wenn eine Task in der Taskkonfiguration eingefügt wird (Befehl 'Objekt hinzufügen'), öffnet der Task Editor-Dialog 'Konfiguration' zum Einstellen der Task-Eigenschaften:

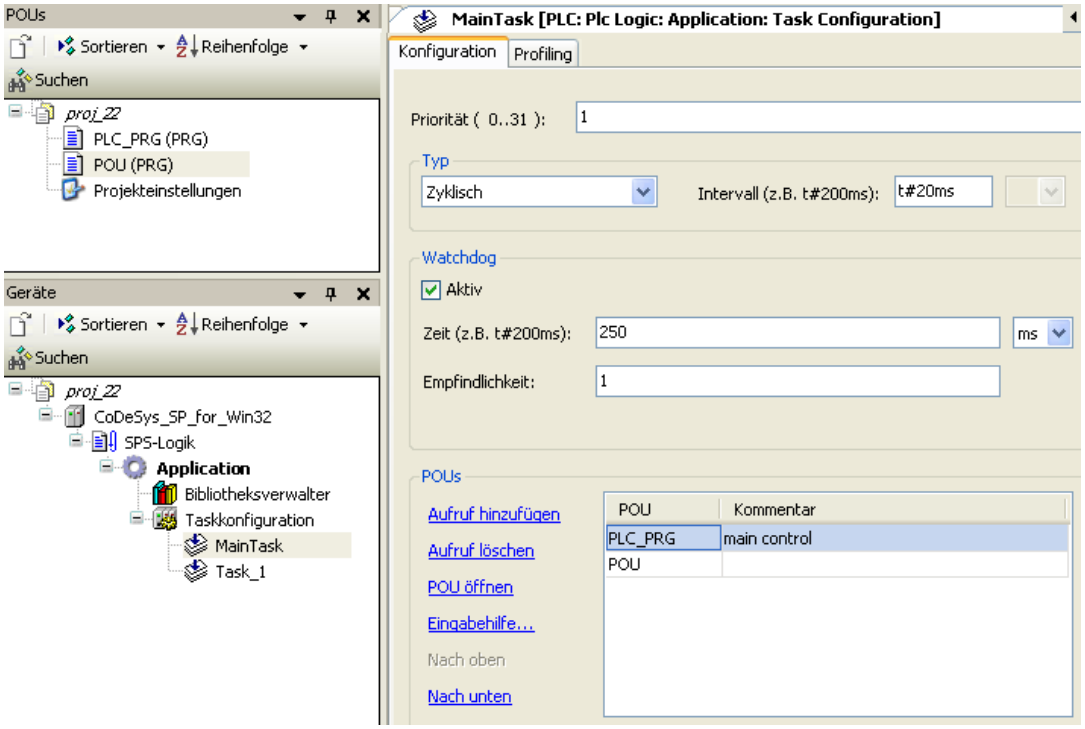


Abb. 413 Konfigurationsdialog für eine Task

Definieren Sie die Parameter wie gewünscht:

Hinweis: Eine Task kann durch Editieren des Namens im Gerätebaum umbenannt werden.

Prioriät (0-31): (Zahl zwischen 0 und 31; 0 ist die höchste, 31 die niedrigste Priorität),

Typ: Die Auswahlliste bietet die folgenden Task-Typen:

Zyklisch: Die Task wird zyklisch abgearbeitet, mit einer Zykluszeit wie im Feld Intervall definiert (s.u.).

Freilaufend: Die Abarbeitung der Task bei Programmstart gestartet und am Ende eines Durchlaufs wird automatisch in einer fortlaufenden Schleife neu gestartet. Es ist keine Zykluszeit definiert.

Status: Die Abarbeitung der Task wird gestartet, wenn die Variable, die im Feld 'Ereignis' definiert ist, den booleschen Wert TRUE hat.

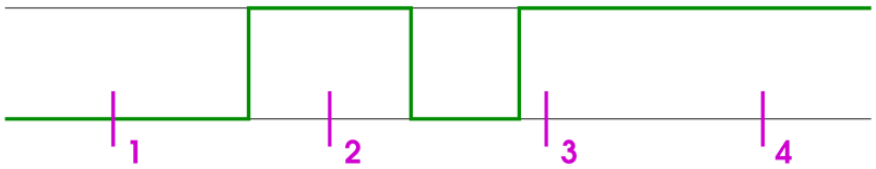
Ereignis: Die Abarbeitung der Task wird gestartet, sobald die Variable, die im Feld 'Ereignis' definiert ist, eine steigende Flanke erhält.

Externes Ereignis: Die Abarbeitung der Task wird gestartet, sobald das Ereignis, das im Feld 'Ereignis' definiert ist, eintritt. Es hängt vom Zielsystem ab, welche Ereignisse unterstützt und in der Auswahlliste angeboten werden. (Nicht zu verwechseln mit Systemereignissen)

Unterschied zwischen Status und Ereignis:

Hat das festgesetzte Ereignis den Wert TRUE, so ist die Startbedingung einer Task vom Typ "Status" erfüllt. Im Gegensatz dazu verlangt der Start einer Task vom Typ "Ereignis" einen Wechsel des Ereignisses von FALSE auf TRUE. Ist die Abtastrate des Task Scheduler zu niedrig, können steigende Flanken des Ereignisses allerdings unbemerkt bleiben.

Das folgende Beispiel illustriert das unterschiedliche Verhalten einer Task in Reaktion auf ein Ereignis (grüner Graph):



An den Abtastpunkten 1-4 (magenta) zeigen Tasks unterschiedlichen Typs eine unterschiedliche Reaktion auf das Ereignis:

Verhalten am Punkt:	1	2	3	4
Status	kein Start	Start	Start	Start
Event	kein Start	Start	kein Start	kein Start

Obligatorische Einträge in Abhängigkeit vom Typ der Task:

Intervall (obligatorischer Eintrag für Task-Typ 'zyklisch' bzw. für 'Externes Ereignis', wenn das Ereignis eine Zeitangabe erfordert): Zeitspanne, nach der die Task neu gestartet werden soll. Wenn hier eine Zahl angegeben wird, kann in der Auswahlbox hinter dem Eingabefeld die gewünschte Einheit eingestellt werden: Millisekunden [ms] oder Mikrosekunden [µs]. Wenn das [ms]-Format gewählt wird, wird eine Eingabe automatisch im TIME-Format dargestellt (e.g. "t#200ms"), sobald das Fenster erneut den Fokus erhält; es kann aber auch direkt im TIME-Format eingegeben werden. Eingaben im [ms]-Format werden immer als reine Zahl dargestellt (z.B. "300").

Ereignis (obligatorischer Eintrag für Task-Typ 'Ereignis' oder 'Externes Ereignis'): Globale Variable, die den Start der Taskabarbeitung triggert, sobald sie eine steigende Flanke erhält. Verwenden Sie Schaltfläche **...** oder die Eingabehilfe <F2> um eine Liste aller verfügbaren globalen Variablen zu erhalten.

Note: Stammt das Ereignis, das eine Task steuert, aus einem Eingang, so muss es zumindest eine Task geben, die nicht ereignisgesteuert ist. Ansonsten werden die Ein-und Ausgänge nie aktualisiert und die ereignisgesteuerte Task wird nie gestartet.

NOCH NICHT IMPLEMENTIERT: Möglicherweise definiert das Zielsystem Singleton Events. Dies sind Ereignisse, die nur das Starten einer einzigen Task erlauben. Ob mehrfach gestartet wird, wird während des Kompilierens geprüft. Die Prüfung berücksichtigt die Datenadresse der Ereignis-Variable, nicht deren Namen. Beispiel: Wenn das Zielsystem %MX1.1 und %IB4 als Singleton-Events definiert, werden zwei Fehler ausgegebn, wenn die folgenden Variablen als Ereignis-Variablen verwendet werden (wobei a und b wie auch c und d jeweils die selbe Adresse haben).

```
VAR_GLOBAL
a AT %MX1.1: BOOL;
b AT %MX1.1: BOOL;
c AT %MB4: BOOL;
d AT %MD1: BOOL;
END_VAR
```

Watchdog

Für jede Task kann eine Zeitüberwachung (Watchdog) konfiguriert werden. Wenn das Zielsystem eine "erweiterte" Watchdog-Konfiguration unterstützt, sind möglicherweise obere und untere Grenzen, eine Default-Watchdog-Zeit wie auch eine Zeitangabe in Prozent in der Gerätebeschreibung vordefiniert.

Aktiv: Wenn diese Option aktiviert ist (), ist der Watchdog aktiv. Das bedeutet, dass die Task mit Fehler-Status abgebrochen ("Ausnahmefehler") wird, sobald die Abarbeitung länger dauert als die aktuell eingestellte **Zeit** (siehe unten), wobei die aktuell angegebene **Empfindlichkeit** (siehe unten) mit eingerechnet wird. Ist die Option "Im Stop-Status E/A aktualisieren" in den SPS-Einstellungen aktiviert, so werden die Ausgänge auf die definierten Standardwerte zurückgesetzt.

Die folgenden Fälle sind möglich:

1. Mehrere aufeinanderfolgende Zeitüberschreitungen; folgendes gilt:

Empfindlichkeit	Ausnahmefehler in Zyklus...
0, 1, 2	1
3	2
...	...
n	n-1

2. Einmalige Zeitüberschreitung: Ausnahmefehler, wenn die Zykluszeit des aktuellen Zyklus länger als (Zeit * Empfindlichkeit) ist. Beispiel: Zeit=t#10ms, Empfindlichkeit=5 -> Ausnahmefehler sobald die Task (einmalig) länger als 50ms läuft.

Zeit (z.B. t#200ms): Watchdog-Zeit; Bedeutung siehe oben: Aktiv. Abhängig vom Zielsystem muss die Überwachungs-Zeitspanne ggfs. in Prozent des Task-Intervalls angegeben werden. In diesem Fall ist die Auswahlbox für die Einheit gegraut dargestellt und zeigt "%" an.

Empfindlichkeit: Zahl; Bedeutung siehe oben: Aktiv.

Hinweis: In der Initialisierungsphase direkt nach einem Neustart der PLC Applikation treten einzelne PLC Zyklen mit verlängerter Laufzeit auf. Damit der Watchdog hierbei nicht anspricht, sollte die folgende Parametrierung gewählt werden:

Zeit : doppelte Zykluszeit der zu überwachenden Task

Empfindlichkeit : 3

Beachten Sie, dass ein Watchdog mit Hilfe der Bibliotheksfuntionen CmpIecTask.library für bestimmte SPS Zyklen abgeschaltet werden kann; das ist nützlich für Zyklen, die aufgrund von Initialisierungen mehr Zeit beanspruchen können.

Nach Deklaration einer geeigneten Variablen für das Taskhandle (vom Typ RTS_IEC_HANDLE),

```
hIecTask : RTS_IEC_HANDLE;
```

kann das Deaktivieren (und anschließende Reaktivieren) des Watchdogs folgendermaßen mit Hilfe der Schnittstellenfunktionen realisiert werden:

```
hIecTask := IecTaskGetCurrent(0);
IecTaskDisableWatchdog(hIecTask);
... // Code that is protected against watchdog
IecTaskEnableWatchdog(hIecTask);
```

POUs:

Die Bausteine, die von der Task kontrolliert werden, sind hier in einer Tabelle aufgelistet, mit Bausteinname (**POU**) und optional einem **Kommentar**. Links der Tabelle finden sich Befehle zur Bearbeitung:

- Um einen neuen Programmaufruf zu definieren, öffnen Sie mit Befehl **Aufruf hinzufügen** den Eingabehilfe-Dialog, und wählen aus den im Projekt verfügbaren das gewünschte Programm aus.
- Um einen Aufruf durch einen anderen zu ersetzen, markieren Sie den Eintrag in der Tabelle, öffnen die Eingabehilfe mit Befehl **Eingabehilfe** und wählen ein anderes Programm aus.
- Um einen Eintrag zu löschen, markieren Sie ihn in der Tabelle und verwenden Befehl **Aufruf löschen**.
- Mit Befehl **Baustein öffnen** wird das gerade ausgewählte Programm im zugehörigen Editor geöffnet.

Die Anordnung der Bausteine in der Liste von oben nach unten bestimmt die **Ausführungsreihenfolge** der Programme im Online-Betrieb. Mit den Befehlen **Nach oben** und **Nach unten** kann der gerade ausgewählte Eintrag innerhalb der Liste verschoben werden.

NOCH NICHT IMPLEMENTIERT: Herstellerspezfiische Attribute:

Zusätzlich zu den Standardeinstellungen sind möglicherweise herstellerspezifische in einem weiteren Registerblatt "**Parameter**" verfügbar. Dies ist der Fall, wenn entsprechende Definitionen in der target-spezifischen Beschreibungsdatei vorhanden sind..

Dialog 'Parameter'

Dieser Dialog ist nur dann als weiteres Registerblatt im Task-Editor verfügbar, wenn dies vom Zielsystem so definiert ist. Er ziegt herstellerspezifische Attribute der Task an (Name, Typ, Wert, Beschreibung).

Dialog 'Systemereignisse'

NOCH NICHT VERFÜGBAR

13.16.2 Task Editor im Online-Modus

Welche Task wird abgearbeitet

Für die Abarbeitung der in der Taskkonfiguration definierten Tasks gelten folgende Regeln:

- Eine Task wird abgearbeitet, wenn ihre Bedingung erfüllt ist, d.h. wenn die angegebene Zeit abgelaufen ist oder die angegebene Variable (Ereignis) eine steigende Flanke erhält.
- Wenn mehrere Tasks gleichzeitig die Bedingung zur Abarbeitung erfüllen, wird zuerst diejenige mit der höchsten Priorität abgearbeitet.
- Wenn mehrere Tasks gleichzeitig die Bedingung zur Abarbeitung erfüllen und gleiche Prioritätsstufe haben, wird zuerst diejenige, die bereits am längsten gewartet hat, abgearbeitet.
- Die Abarbeitung der Programmaufrufe erfolgt jeweils in der Reihenfolge, in der sie (von oben nach unten) im Konfigurationsdialog der Task angeordnet sind. Wenn ein Programm aufgerufen wird, das mit gleichem Namen sowohl im Gerätebaum bei der Applikation als auch in einer Bibliothek oder projektglobal im POUs-Fenster vorliegt, wird das direkt der Applikation zugeordnete verwendet.

Monitor, Online-Ansicht des Task Editors

Wenn der oberste Knoten im Baum der Taskkonfiguration ausgewählt ist, steht neben dem Eigenschaften-Dialog als weiteres Tabellenblatt der Dialog 'Überwachungr' bereit. Er zeigt im Online-Modus für die in der Taskkonfiguration enthaltenen Tasks jeweils den Status und einige aktuelle Messungen zu den Zyklen und Zykluszeiten in einer Tabellenansicht dar. Die Werte werden im gleichen Zeitintervall aktualisiert wie beim Monitoring von Werten aus der Steuerung.

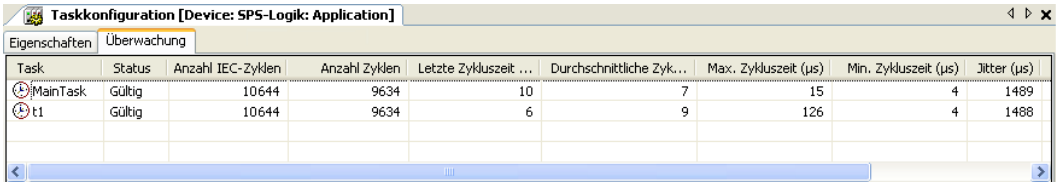


Abb. 414 Taskkonfiguration, Überwachung

Für jede Task werden die folgenden Informationen in einer Zeile angezeigt:

Task	**Taskname, wie in der Taskkonfiguration definiert**
Status	mögliche Angaben: Not created: Task wurde seit der letzten Aktualisierung noch nicht gestartet; speziell für Ereignistasks Created: Task ist im Laufzeitsystem bekannt, wurde aber noch nicht für den Betrieb aufgesetzt Valid: Task ist normal in Betrieb Exception: Task hat Ausnahmezustand
Anzahl IEC-Zyklen	Anzahl der seit dem Starten der Applikation ausgeführten Zyklen; "0", wenn die Funktion vom Zielsystem nicht unterstützt wird.
Anzahl Zyklen	Anzahl bereits ausgeführter Zyklen; (abhängig vom Zielsystem kann dies der Anzahl IEC-Zyklen entsprechen, oder größer sein, wenn auch gezählt wird während die Applikation nicht läuft.
Letzte Zykluszeit (µs)	Letzte gemessene Zykluszeit in µs
Durchschnittliche Zykluszeit (µs)	Durchschnittliche Zykluszeit über alle Zyklen in µs
Max. Zykluszeit (µs)	Maximale gemessene Zykluszeit über alle Zyklen in µs
Min. Zykluszeit (µs)	Minimale gemessene Zykluszeit über alle Zyklen in µs
Jitter (µs)	Letzter gemessener Jitter* in µs
Min. Jitter (µs)	Minimaler gemessener Jitter* in µs
Max. Jitter (µs)	Maximaler gemessener Jitter* in µs

* Jitter: Zeit zwischen dem Start der Task und dem Zeitpunkt, ab dem das Betriebssystem anzeigt, dass sie läuft.

Wenn der Cursor auf einem Tasknamen-Feld platziert ist, können die angezeigten Werte mit Befehl **Zurücksetzen** aus dem Kontextmenü auf 0 zurückgesetzt werden.

13.17 Trace-Editor

Die "Trace"-Funktionalität wird mit dem Plug-In 'Trace Editor' bereitgestellt. Sie erlaubt das Aufzeichnen des Verlaufs von Variablenwerten aus der Steuerung über einen bestimmten Zeitraum. Zu diesem Zweck werden die Werte definierter Trace-Variablen kontinuierlich in einen »PLC Designer« Puffer von bestimmter Größe geschrieben und können dann in Form einer Kurve bzw. eines Graphen über einer Zeitachse in »PLC Designer« angezeigt werden.

Die Konfiguration und die Darstellung einer Trace-Aufzeichnung erfolgt in Dialogen und Fenstern des Trace-Editors.

Um die Trace-Aufzeichnung gemäß der aktuellen Trace-Konfiguration im Online-Modus zu starten, muss erst die **Konfiguration ins Laufzeitsystem geladen** werden. Die Kurven der Trace-Variablen werden dann im Trace-Fenster dargestellt und können bei Bedarf in eine externe Datei gespeichert und aus dieser wieder in den Editor geladen werden.

Befehle für das Arbeiten in den Konfigurationsdialogen wie auch für das Beeinflussen der Graphen sind im Editor verfügbar.

Der Trace kann in eine Visualisierung eingebunden werden mit Hilfe des speziellen Visualisierungselements "Trace".

13.17.1 Trace-Editor

Auf dieser Seite zu sehen:

- Überblick
- Befehle und Zoomen im Trace-Fenster
- Konfiguration des Traces
- Trace-Editor im Online-Modus

Überblick

Der Trace-Editor, bereitgestellt mit dem Plug-In 'Trace Editor', dient dem Konfigurieren und Anzeigen von Trace-Aufzeichnungen. Eine "Trace-Aufzeichnung" ist die **Aufzeichnung der Werte einer oder mehrerer Variablen**, die zu diesem Zweck über einen gewissen Zeitraum aus der Steuerung gelesen werden. Zusätzlich kann ein Trigger gesetzt werden, was zur Folge hat, dass nur zum Zeitpunkt des Triggers Daten der Variablen aufgezeichnet und angezeigt werden. Die Werte werden im »PLC Designer« Trace-**Puffer** gespeichert und können aus diesem zur Anzeige im Trace-Editor (Trace-Fenster) abgerufen werden. Die Anzeige erfolgt in einem Koordinatensystem in Form einer Kurve über einer Zeitachse. Die Darstellungsform kann konfiguriert werden.

Existiert im Laufzeitsystem die zusätzliche Komponente 'Trace Manager', werden dort die Trace-Konfigurationen verwaltet und die Daten gespeichert. Außerdem besteht mit Hilfe des Trace-Managers die Möglichkeit, auf gerätespezifische Traces zuzugreifen, die z.B. Stromwerte aufzuzeichnen. Steht der Trace-Manager nicht zur Verfügung, wird die Funktionalität über IEC-Code heruntergeladen. Die Menü-Befehle **Trace hochladen** und **Online-Liste....** stehen dann nicht zur Verfügung.

Variablen, die simultan getriggert werden, können im gleichen Trace-Objekt aufgezeichnet werden. Die Darstellung in einem variablen-spezifischen Koordinatensystem im Trace-Fenster kann auch im Multi-Kanal-Modus erfolgen. Traces von Variablen mit unterschiedlichen Triggern werden in jeweils eingenen Trace-Objekten aufgezeichnet. Es dürfen beliebig viele Trace-Objekte angelegt werden.

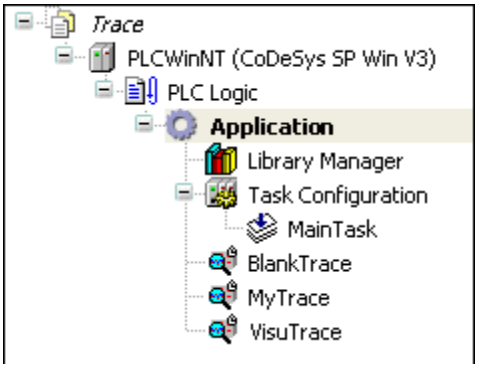


Abb. 415 Beispiel: Projekt mit Traces

Die folgende Abbildung zeigt eine Trace-Aufzeichnung im Online-Betrieb. Vier Aufzeichnungen wurden im "Trace-Baum" der verfügbaren Aufzeichnungen im rechten Teil des Trace-Windows zur Anzeige ausgewählt. Im Kontextmenü des Trace-Baums stehen die Befehle zum Hinzufügen oder Löschen von Variablen, um sie sichtbar zu machen, um ihre Anzeigeeinstellungen zu ändern, und zum Öffnen des Konfigurationsdialogs.

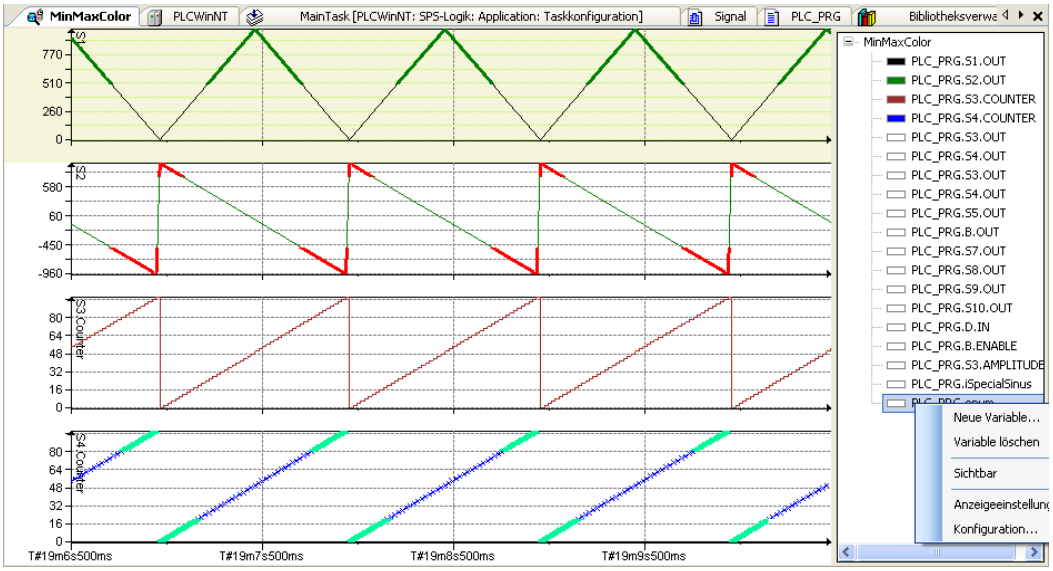


Abb. 416 Beispiel: Trace-Editor-Fenster, Anzeige eines Traces mit vier Variablen im Online-Modus

Befehle und Zoomen im Trace-Fenster

Für das Arbeiten im Trace-Editor im Offline- bzw. Online-Modus stehen Befehle im Trace-Fenster wie auch im Trace Menü bereit. Das Trace-Menü stellt die Befehle zum Bedienen des Trace-Graphen zur Verfügung. Die Menübefehle werden ebenfalls im Kontextmenü des Trace-Diagramms angezeigt. Die Trace-Konfiguration sowie die Trigger-Einstellung und die Variablenauswahl kann nur im Kontextmenü des rechten Teils des Trace-Fensters vorgenommen werden, wo sich der Trace-Baum befindet. Entsprechend der Selektion im Baum werden die Kommandos freigegeben.

Beachten Sie, dass neben den "Zoom"-Befehlen generell auch eine "Scroll-Maus" verwendet werden kann, um im Trace-Fenster zu **zoomen**.

Konfiguration des Traces

Ein "Trace"-Objekt kann im Gerätebaum unterhalb einer Applikation mit Hilfe des Befehls 'Objekt hinzufügen' eingefügt werden und mit Befehl 'Objekt bearbeiten' oder einem Doppelklick auf das Objekt kann der Trace-Editor geöffnet werden.

Zunächst zeigt das Trace-Fenster, betitelt mit "**<Trace-Objekt-Name>**" im linken Teil einen leeren Bereich, in dem später der Trace-Graph angezeigt wird, im rechten Teil den Trace-Baum, so wie er in Konfiguration... festgelegt ist.

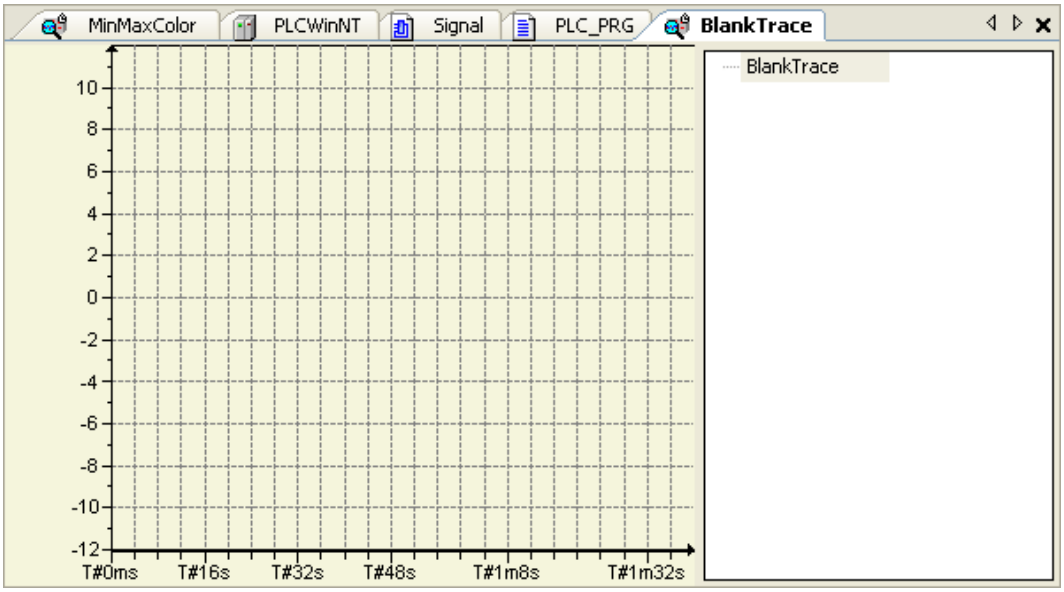


Abb. 417 Trace-Editor-Fenster, noch ohne benutzerspezifische Konfiguration

Im Trace wird mindestens eine Variable festgelegt, die aufgezeichnet wird. Dazu werden allgemeine Einstellungen definiert, die den Start der Aufzeichnung, der Abtastung, die Trigger-Bedingungen, die Puffergröße und die zuständige Task usw. festlegen. Außerdem werden hier die Anzeigeeinstellungen (Anzeigeparameter des Koordinatensystems, in dem der Graph angezeigt wird) für jede dieser Trace-Variablen gesetzt.

Um einen Trace zu konfigurieren, verwenden Sie im Kontextmenü des rechten Teils des Trace-Fensters ...

1. Neue Variable...: um eine Variable hinzuzufügen und um einige Anzeigeeinstellungen zu setzen.
2. **Variable löschen:** um selektierte Variablen zu löschen
3. **Sichtbar:** um Variablen auf sichtbar zu setzen.
4. Anzeigeeinstelllungen...: um die Anzeigeeinstellungen des Graphen festzulegen. (Solange keine Konfiguration geladen ist, ist dieser Befehl gegrayt.)
5. Konfiguration...: um den Trace-Graphen zu definieren (Bedingungen, Kurventypen, und nochmals die Anzeigeeinstellungen) mit seinen speziellen Trace-Variablen.

Eine Trace-Konfiguration kann **in eine externe Datei *.trace gespeichert** und aus dieser wieder ins Trace-Fenster geladen werden.

Trace-Editor im Online-Modus

Um eine Trace-Aufzeichung durchführen zu können, muss der für den Trace erzeugte Code zuerst explizit ins Zielsystem **geladen** werden.

Das **'Trace'-Menü** enthält zusätzliche **Befehle** für das Arbeiten in den im Online-Modus dargestellten Aufzeichungskurven.

Neue Variable...

Mit diesem Kommando, das im Kontextmenü des Trace-Baumes im Trace Editor zur Verfügung steht, öffnet der Konfigurationsdialog mit den Variableneinstellungen.

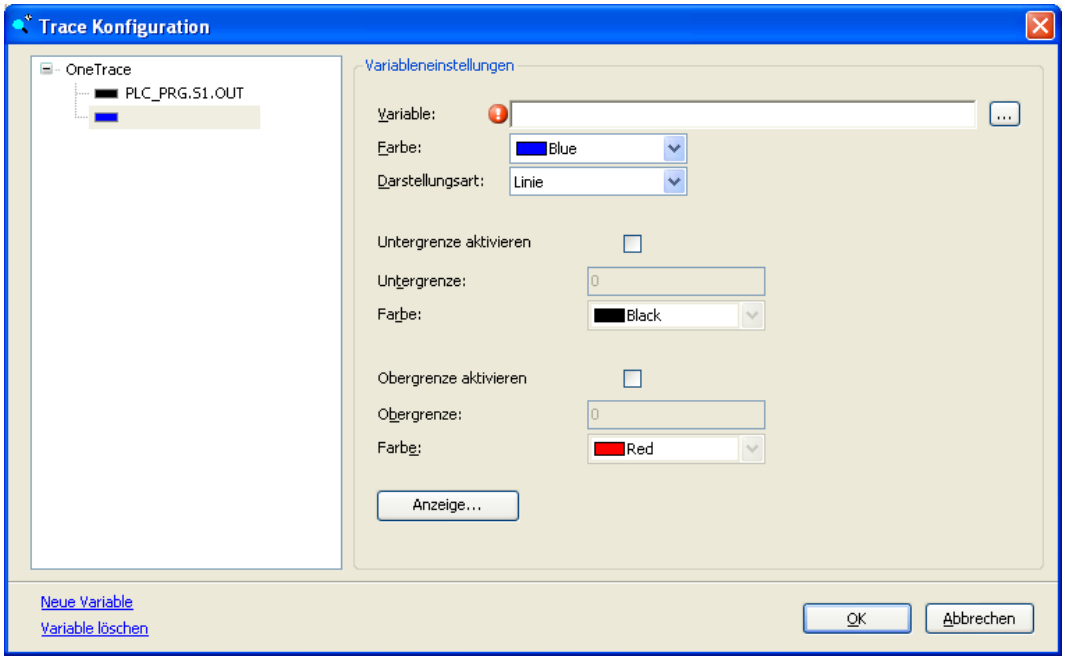


Abb. 418 Trace-Konfigurationsdialog

Variable: Geben Sie den Namen (Pfad) der Variable ein, deren Werte in der betreffenden Aufzeichnung erfaßt werden sollen. Mit ... kann die Eingabehilfe geöffnet werden, was die Eingabe von gültigen Variablen erleichtert. Die Variable wird unmittelbar im Konfigurationsbaum (Tracebaum) angezeigt. Die aktuellen Variableneinstellungen können jederzeit später wieder bearbeitet werden, wenn der Eintrag im Konfigurationsbaum selektiert ist.

Farbe: Wählen Sie aus der Farbauswahlliste eine Farbe für die Wertekurve der Variablen.

Darstellungsart: Geben Sie hier an, wie die einzelnen Werte der Variablen in eine Kurve umgesetzt werden sollen. Es wird empfohlen, für große Datenmengen den Typ 'Linie' zu verwenden.

Keine:	Die Werte werden nicht angezeigt.
Linie:	Die einzelnen Werte werden mit einer Linie verbunden.
Kreuze:	Die einzelnen Werte werden als Kreuzchen dargestellt.
Spline:	Die einzelnen Werte werden zu einer Spline-Kurve verbunden.
Stufen:	Die einzelnen Werte werden in Treppenform dargestellt: vom letzten Punkt zieht eine vertikale Linie zum y-Wert des nächsten und von dort eine zu horizontale zu dessen x-Wert.
Punkte:	Die einzelnen Werte werden als Punkt dargestellt.

Untergrenze aktivieren : Wenn diese Option aktiviert ist, wird die Wertekurve mit der unterhalb 'Untergrenze' definierten Farbe dargestellt, sobald die Variable die definierte Untergrenze unterschreitet.

Untergrenze : Wenn der hier eingegebene Wert der Variablen unterschritten wird, und Option 'Untergrenze aktivieren' aktiviert ist, wechselt die Wertekurve in die folgend angegebene Farbe.

Farbe : s.o. 'Untergrenze aktivieren'

Obergrenze aktivieren: Wenn diese Option aktiviert ist, wird die Wertekurve mit der unterhalb 'Obergrenze' definierten Farbe dargestellt, sobald die Variable die definierte Obergrenze überschreitet.

Obergrenze: Wenn der hier eingegebene Wert der Variablen unterschritten wird, und Option 'Obergrenze aktivieren' aktiviert ist, wechselt die Wertekurve in die folgend angegebene Farbe.

Farbe: s.o. 'Obergrenze aktivieren'

Konfiguration...

Auf dieser Seite zu sehen:

- Aufzeichnungseinstellungen
- Variableneinstellungen

Der Dialog **Trace Konfiguration** öffnet sich, wenn der Befehl **Konfiguration...** im Kontextmenü des Tracebaumes im rechten Teil des Trace Editors selektiert wird oder nach Doppelklick auf den Trace-Namen oder den Variablennamen. Hier wird konfiguriert, mit welchen Einstellungen getracet wird oder welche Variablen mit welchen Parametern aufgezeichnet werden sollen.

Hinweis: Die im Dialog Aufzeichnungeinstellungen gesetzten Parameter gelten für alle Variablen des Trace-Graphen.

Die Trace-Konfiguration besteht aus einer Trace-Aufzeichnungskonfiguration (Aufzeichnungseinstellung) und aus der Variablenkonfiguration (Variableneinstellung). Die konfigurierten Trace-Variablen sind im linken Teil des Trace-Fensters in einer Baumstruktur, unter dem obersten Knoten mit Namen des Traces, dargestellt. Dies wird auch im Hauptfenster des Trace-Editors dargestellt.

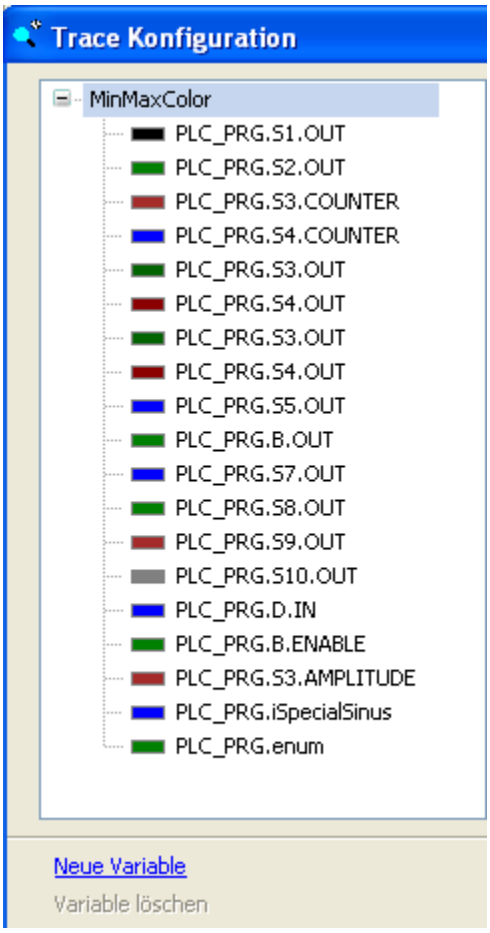


Abb. 419 "Trace-Baum" des Konfigurationsdialogs

Ist der Name des Traces im Trace-Baum selektiert, ist der Dialog mit den Aufzeichnungseinstellungen offen und diese können geändert werden. Um eine Variable hinzuzufügen oder um eine zu löschen stehen folgende Befehle unterhalb des Trace-Baumes bereit: **Neue Variable**, **Variable löschen**.

Um die bestehende Variableneinstellung zu ändern ist die betroffene Variabel zu selektieren um den Dialog zur Variableneinstellung zu öffnen.

Daneben ist es möglich mit **Umschalttaste + Mausklick** oder **Strg + Mausklick** mehrere Variablen zu selektieren. Dann ist die Änderung im Dialog der Variableneinstellung für alle selektierten Variablen durchgeführt.

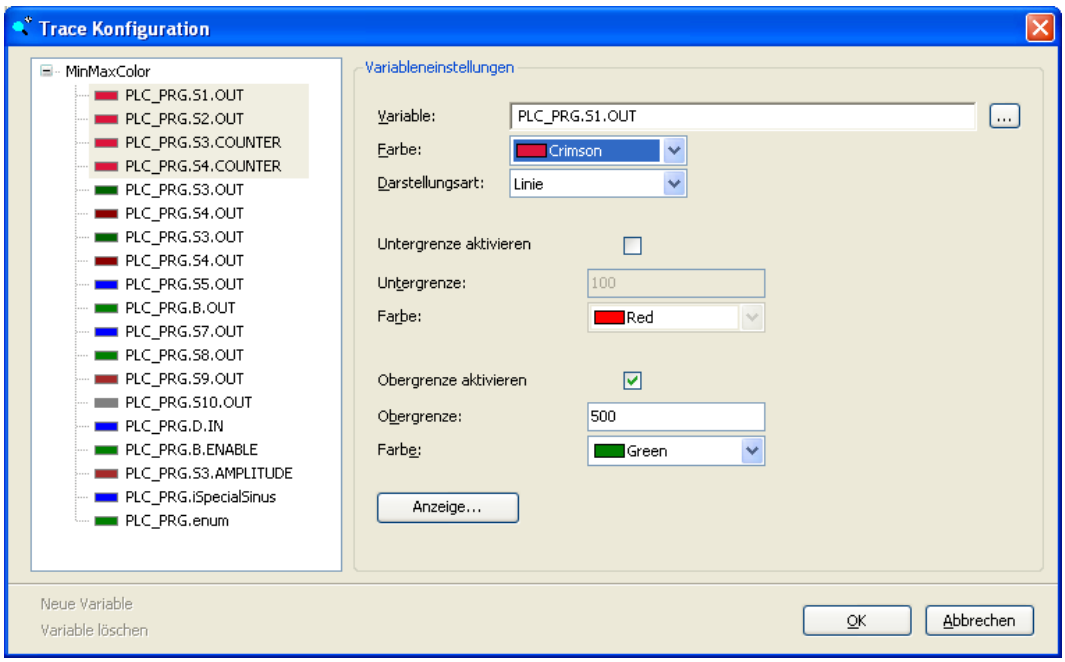


Abb. 420 Multiselektion von Variablen

Um die Einstellungen oder Eigenschaften der Variablen in einem Schritt zu ändern, sind diese über Multiselektion mittels **Umschalttaste** oder **Strg + Mausklick** zu selektieren.

Aufzeichnungseinstellungen

Im folgenden ist eine Beschreibung des Dialogs der Aufzeichnungseinstellungen zu sehen:

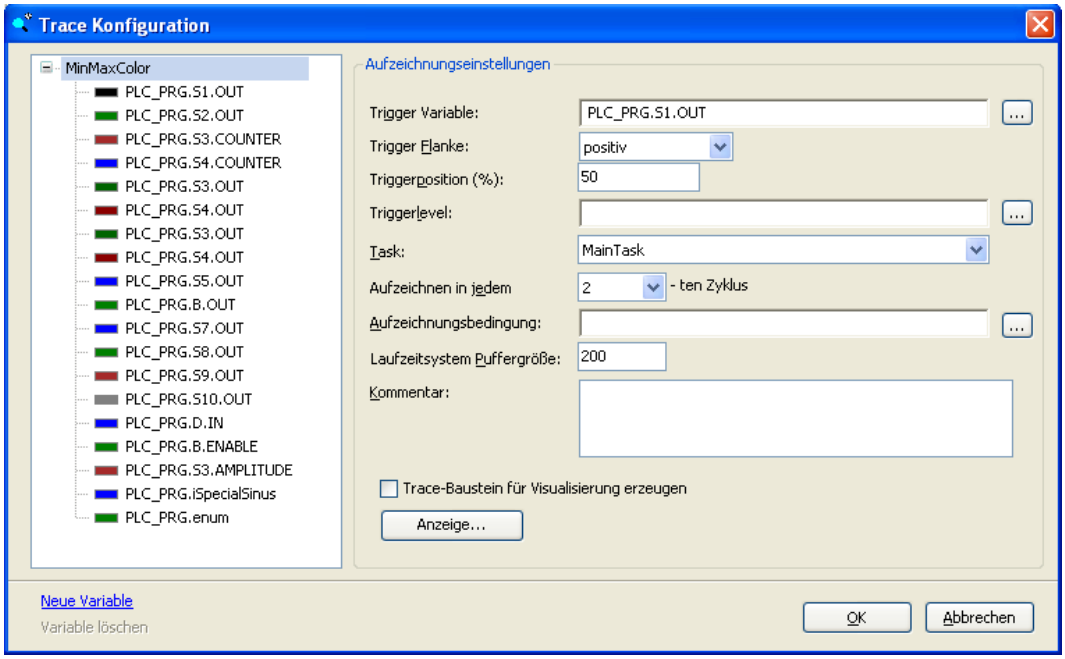


Abb. 421 Trace Konfiguration, Aufzeichnungseinstellungen

Triggervariable: Optional kann ein Trigger konfiguriert werden, der die Zeitspanne der Aufzeichnung gemäß bestimmter Bedingungen festlegt. Die folgenden Einstellungen müssen zu diesem Zweck vorgenommen werden:

Geben Sie eine boolesche Variable, einen Ausdruck oder eine analoge Variable ein. Ebenfalls eingegeben werden können Enumerationsvariablen oder Variablen von Eigenschaften. Wenn diese Variable bzw. dieser Ausdruck einen bestimmten Wert hat (der unten eingestellte Triggerflankentyp ist zu beachten), wird die Aufzeichnung - aber erst nachdem zusätzlich der unten im Feld Trigger Position eingestellte Prozentsatz von Werten gesammelt wurde ! - beendet. Das bedeutet, dass sobald der Trigger aktiv wird, die Werteaufzeichnung noch über eine bestimmte Anzahl von Zyklen weiterläuft.
Über Schaltfläche [...] kann die Eingabehilfe zum Eintragen einer geeigneten Trigger-Variable geöffnet werden.

Beachten Sie die Möglichkeit, mit Hilfe eines Eintrags bei 'Aufzeichungsbedingung' (s.u.) auch den Startzeitpunkt der Werteaufzeichnung zu kontrollieren.

Triggerflanke:

keine	kein Trigger gesetzt (Default)
positiv	Triggerereignis bei einer steigenden Flanke auf die boolsche Triggervariable oder - bei einer analogen Triggervariablen - sobald der in 'Triggerlevel' definierte Wert "von unten" her erreicht wird.
negativ	Triggerereignis bei einer steigenden Flanke auf die boolsche Triggervariable oder - bei einer analogen Triggervariablen - sobald der in 'Triggerlevel' definierte Wert "von oben" her erreicht wird.
beide	Triggerereignis bei den für 'positiv' und 'negativ' (s.o.) beschriebenen Bedingungen.

Triggerposition (%): Tragen Sie hier den Prozentsatz aller aufgezeichneten Werte einer Trace-Variablen ein, der **nach** dem Auftreten des Trigger-Ereignisses aufgezeichnet werden soll. Das heißt, hier kann festgelegt werden, welcher Teil der Werte vor, und welcher nach dem Trigger-Ereignis aufgezeichnet werden soll. Beispiel: Wenn hier "25" eingetragen wird, werden in der Wertekurve 75 % der gemessenen Werte vor dem Zeitpunkt des Trigger-Ereignis dargestellt und 25% danach. Wenn die Aufzeichnungskurve genau beim Trigger-Ereignis starten soll, muss hier 100% eingegeben werden.

Triggerlevel: Wenn eine analoge Variable als Trigger Variable angegeben ist, muss hier der Wert eingetragen werden, bei dem diese das Trigger-Ereignis auslöst. Geben Sie den Wert direkt oder über eine Variable, die den Wert liefert, an (auch ENUM-Konstanten sind erlaubt). Default: kein Eintrag.

Task: Von der Liste der verfügbaren Tasks wählen Sie diejenige, nach deren Abarbeitung der Wert der Trace-Variablen gelesen werden soll. Default: Erste Task im Task Konfigurationsbaum.

Aufzeichnen in jedem ...- ten Zyklus: Geben Sie die Anzahl der Zyklen an, die verstreichen sollen, bevor der Wert der Trace-Variablen erneut gelesen werden soll. Mit der Formel (Puffergröße * Aufzeichnen in jedem x-ten Zyklus * Task-Intervall) kann die Mindest-Aufzeichnugszeitspanne abgeschätzt werden.

Aufzeichnungsbedingung: Eine boolsche Variable, ein Wert oder ein boolscher Ausdruck kann hier eingetragen werden. Sobald diese Bedingung eintritt, also Wert TRUE erhält, wird die Werteaufzeichnung gestartet. Wenn hier nichts angegeben ist, startet die Werteaufzeichnung sobald die Trace Konfiguration in die Steuerung geladen wurde und die Applikation läuft.

Puffergröße: Größe des Ringpuffers in Bytes, der im Laufzeitsystem für das Sammeln der Trace-Variablenwerte der gerade konfigurierten Aufzeichnung vorgesehen ist. Der Pufferinhalt wird dann als Wertekurve(n) im Trace-Fenster angezeigt. Default: 100.

Kommentar: Hier kann ein Kommentar-Text zur Aufzeichnung eingegeben werden.

Anzeigeeinstellungen: Diese Schaltfläche öffnet den Dialog **Anzeigeeinstellungen <Aufzeichnung>**, wo die Gestaltung des Trace-Fensters für die gerade konfigurierte Aufzeichnung definiert wird (Achsen, Farben, Scroll-Verhalten).

Trace-Baustein für Visualisierung erzeugen: Die Aktivierung des zugehörigen Kontrollkästchens hat die implizite Erzeugung des Bausteins <Tracename>_<Taskname>_VISU zur Folge. Dies ist Voraussetzung, falls die Trace-Aufzeichnung in eine Visualisierung integriert werden soll.

Hinweis:Wenn verschiedene Anzeige-Arten für dieselbe Trace-Variable bereitstehen sollen, muss die Variable in weiteren Trace-Objekten deklariert werden, die dann unterschiedliche Variablen- oder Anzeigeeinstellungen haben.

Schaltfläche **Anzeige:**

Die folgenden Einstellungen werden für die X-Achse und Y-Achse des Trace-Graphen vorgenommen und sind sichtbar, wenn das Trace-Diagramm in Einkanal-Ansicht (Standardeinstellungt) angezeigt wird.

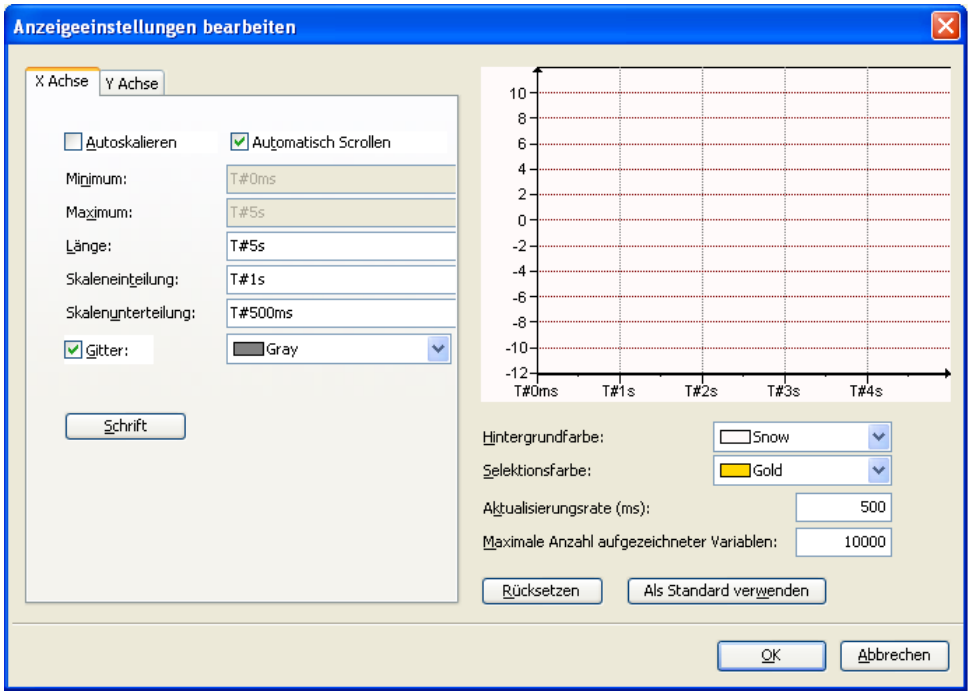


Abb. 422 Dialog 'Anzeigeeinstellungen bearbeiten'

Autoskalieren: Wenn diese Option aktiviert ist, wird die Zeitspanne, die im Trace-Fenster auf der X-Achse angezeigt wird, automatisch der Puffergröße angepaßt, die für die Aufzeichnung im Laufzeitsystem definiert ist.

Wenn die Option nicht aktiviert ist, kann die Zeitspanne, die im Trace-Fenster dargestellt werden soll, durch folgende Einstellungen bestimmt werden:

Wenn die Option **Automatisch Scrollen** nicht aktiviert ist:

Minimum, **Maximum** : Die hier eingegebenen Zeiten (Standardzeitformat, z.B. "T#200s") definieren genau den Zeitraum der Werteaufzeichnungen, der dargestellt werden soll. D.h. die X-Achse beginnt mit der Minimumzeit und endet mit der Maximumzeit. Um einen anderen Zeitbereich der Aufzeichnung sichtbar zu machen, muss händisch gescrollt werden.

Wenn die Option **Automatisch Scrollen** aktiviert ist (kann nur aktiviert werden, wenn die Option **Autoskalieren** deaktiviert ist.)

Länge: Die hier angegebene Zeitspanne (Standardzeitformat, z.B. "T#2m") wird dargestellt, wobei automatisch gescrollt wird, so dass immer die aktuellsten Werte der Aufzeichnung sichtbar sind.

Skaleneinteilung: Unterteilung der Achse mit langen Strichmarkierungen. Wenn die Gitterdarstellung aktiviert ist, werden sie mit gestrichelten Linien verlängert. Die Angabe muss der IEC-Syntax und dem Typ des durch die Achse vertretenen Werts entsprechen, z.B. für die X-Achse ein Zeitwert für den Abstand zwischen den Strichmarken, beispielsweise "T#2s" für 2 Sekunden.

Skalenunterteilung: Unterteilung der Achse mit kurzen Strichmarkierungen, die nicht durch gestrichelte Linien verlängert werden. Gültige Angaben siehe 'Skaleneinteilung'.

Beschreibung: Achsenbeschriftung: Wenn diese Option aktiviert ist, wird der Text, der im Eingabefeld angegeben wird, am oberen Ende der Y-Achse angezeigt. Für die X-Achse kann hier keine Beschriftung angegeben werden.

Gitter: Wenn diese Option aktiviert ist, wird ein Gitternetz im Koordinatensystem eingeblendet. Die Skaleneinteilungen werden zu diesem Zweck mit gestrichelten Linien verlängert. Die Farbe der Gitternetzlinien kann von der Farbauswahlliste gewählt werden.

Schrift: Diese Schaltfläche öffnet den Standarddialog zur Festlegung der Schriftdarstellung im Trace-Fenster.

Hintergrundfarbe: Aus der Farbauswahlliste kann die Hintergrundfarbe für das Koordinatensystem ausgewählt werden, die verwendet werden soll, solange die Aufzeichnung im Trace-Fenster nicht selektiert ist.

Selektionsfarbe: Aus der Farbauswahlliste kann die Hintergrundfarbe für das Koordinatensystem ausgewählt werden, die verwendet werden soll, solange die Aufzeichnung im Trace-Fenster selektiert ist.

Aktualisierungsrate (ms): Zeitintervall in Millisekunden in dem das Update des Trace-Fensters im Online-Modus erfolgen soll. Standardwert: 500ms.

Maximale Anzahl aufgezeichneter Werte: Maximale Anzahl von Werten, die in den »PLC Designer« Trace-Ringpuffer geschrieben werden können. Standardwer: 10000. Ist der Puffer voll, werden die zuerst eingetragenen Werte überschrieben.

Rücksetzen: Mit dieser Schaltfläche werden die Einstellungen des Dialogs auf die Default-Werte zurückgesetzt (nicht zu verwechseln mit den 'Standard'-Einstellungen, siehe im folgenden.

Als Standard verwenden: Mit dieser Schaltfläche werden die aktuellen Einstellungen des Dialogs gespeichert und beim Konfigurieren eines neuen Traces oder einer neuen Variablen bereits vorgegeben.

Variableneinstellungen

Im folgenden ist eine Beschreibung des Dialogs der Einstellungen für die Variableneinstellungen zu sehen:

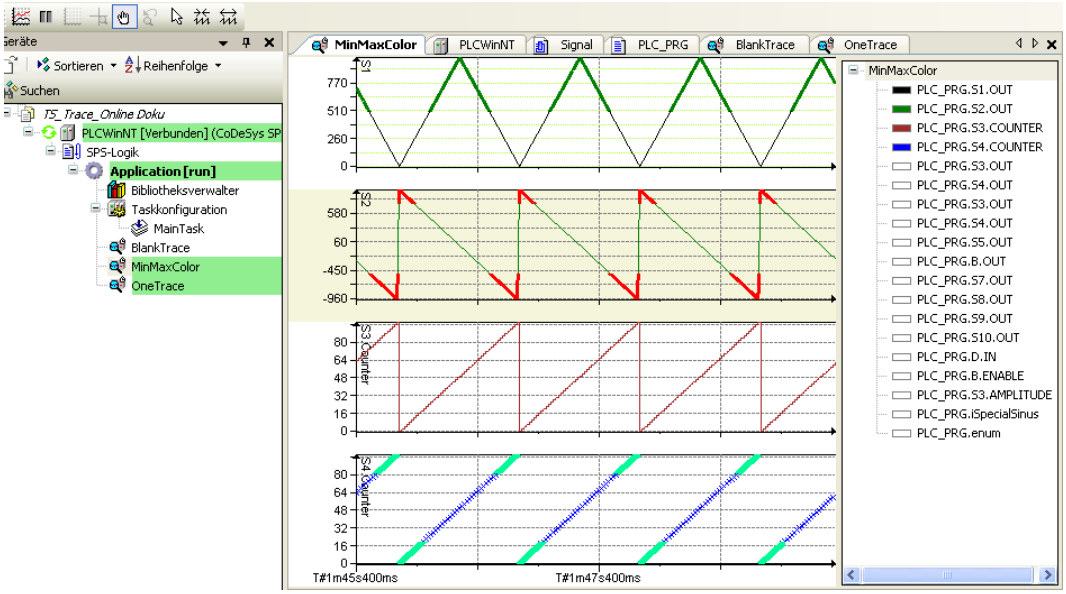


Abb. 423 Trace Konfiguration, Variableneinstellungen

Für jeden Trace müssen die Variablen, deren Werte gesammelt und im Koordinatensystem dargestellt werden sollen, definiert werden. Um eine Variable einem Trace hinzuzufügen, wählen Sie den Befehl **Neue Variable** (unterhalb des Baums). Im rechten Teil des Fensters erscheint daraufhin der Dialog 'Variableneinstellungen' und gleichzeitig erscheint ein leerer Variableneintrag links im Baum eingerückt unterhalb des Traces. Definieren Sie nun die Variable:

Variable: Geben Sie den Namen (Pfad) der Variable ein, deren Werte erfaßt werden sollen. Neben Daten (Variablen) ist es auch möglich den Inhalt von Pointern, Eigenschaften, Referenzen oder Arrayelementen zu tracen. Die Eingabehilfe (...) kann verwendet werden um gültige Eingaben zu erhalten. Prüfen Sie dann bzw. bearbeiten Sie die Variableneinstellungen (s.u.). Die Variable wird unmittelbar im Trace-Baum angezeigt. Die aktuellen Variableneinstellungen können jederzeit bearbeitet werden, wenn der Eintrag im Konfigurationsbaum selektiert ist. Es können auch mehrere Einträge selektiert werden.

Farbe: Wählen Sie aus der Farbauswahlliste eine Farbe für die Wertekurve der Variablen.

Darstellungsart: Geben Sie hier an, wie die einzelnen Werte der Variablen in eine Kurve umgesetzt werden sollen. Es wird empfohlen, für große Datenmengen den Typ 'Linie' zu verwenden.

Keine:	Die Werte werden nicht angezeigt.
Linie:	Die einzelnen Werte werden mit einer Linie verbunden.
Kreuze:	Die einzelnen Werte werden als Kreuzchen dargestellt.
Spline:	Die einzelnen Werte werden zu einer Spline-Kurve verbunden.
Stufen:	Die einzelnen Werte werden in Treppenform dargestellt: vom letzten Punkt zieht eine vertikale Linie zum y-Wert des nächsten und von dort eine zu horizontale zu dessen x-Wert.
Punkte:	Die einzelnen Werte werden als Punkte dargestellt.

Untergrenze aktivieren : Wenn diese Option aktiviert ist, wird die Wertekurve mit der unterhalb 'Untergrenze' definierten Farbe dargestellt, sobald die Variable die definierte Untergrenze unterschreitet.

Untergrenze : Wenn der hier eingegebene Wert der Variablen unterschritten wird, und Option 'Untergrenze aktivieren' aktiviert ist, wechselt die Wertekurve in die folgend angegebene Farbe.

Farbe : s.o. 'Untergrenze aktivieren'

Obergrenze aktivieren: Wenn diese Option aktiviert ist, wird die Wertekurve mit der unterhalb 'Obergrenze' definierten Farbe dargestellt, sobald die Variable die definierte Obergrenze überschreitet.

Obergrenze: Wenn der hier eingegebene Wert der Variablen unterschritten wird, und Option 'Obergrenze aktivieren' aktiviert ist, wechselt die Wertekurve in die folgend angegebene Farbe.

Farbe: s.o. 'Obergrenze aktivieren'

Schaltfläche **Anzeige**: Dieser Button öffnet den Dialog **Darstellung der Y-Achse**, wo für jede Variable in einem eigenen Stil die Darstellung der aktuell konfigurierten Y-Achse (Farbe und Scroll-Verhalten) im Trace-Fensters festgelegt wird. Diese Einstellungen werden benutzt, wenn das Trace-Diagramm in Multikanal-Anzeige dargestellt wird.

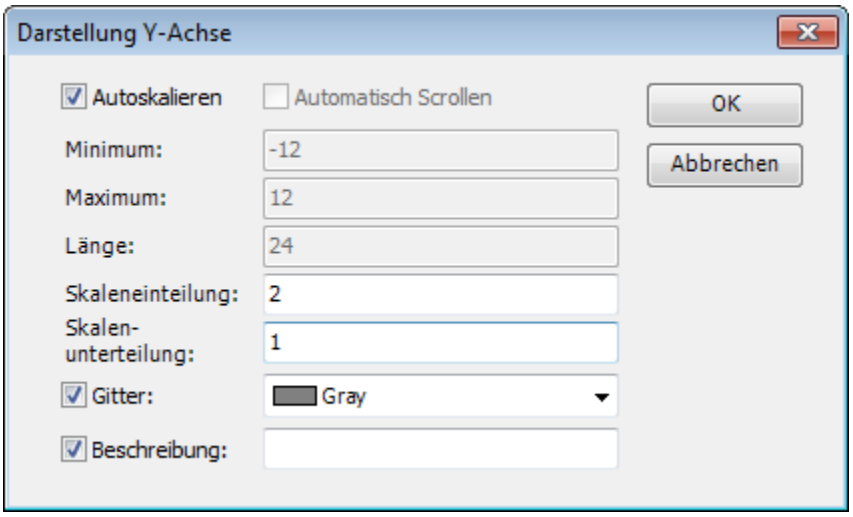


Abb. 424 Editieren des Dialogs 'Darstellung Y-Achse'

Autoskalieren: Ist diese Option aktiviert, dann wird der Trace-Graph automatisch vom minimalen bis zum maximalen Y-Wert vollständig abgebildet.

Ist diese Option nicht aktiviert, dann kann der Ausschnitt der Y-Achse, die im Trace-Fenster angezeigt wird, gemäß folgenden Einstellungen definiert werden:

Ist die Option **Autoscrollen** nicht aktiviert:

Minimum, **Maximum**: Die hier eingegebenen Y-Werte definieren exakt den Ausschnitt des Traces, der anzuzeigen ist. Dies bedeutet, dass der Bereich der Y-Werte mit dem Minimum beginnt und mit dem Maximum endet. Soll ein anderer Bereich angezeigt werden, dann kann manuell gescrollt werden.

Ist die Option **Autoscrollen** aktiviert (nur möglich, wenn die Option **Autoskalieren** nicht aktiviert ist):

Länge: Der hier angegebene Wert legt die Länge des Bereichs fest, der angezeigt wird. Die Achse wird dabei auf eine Weise gescrollt, dass die maximalen Werte sichtbar sind.

Skaleneinteilung: Unterteilung der Achse mit langen Strichmarkierungen. Wenn die Gitterdarstellung aktiviert ist, werden sie mit gestrichelten Linien verlängert. Die Angabe muss der IEC-Syntax und dem Typ des durch die Achse vertretenen Werts entsprechen, z.B. 1000.

Skalenunterteilung: Unterteilung der Achse mit kurzen Strichmarkierungen, die nicht durch gestrichelte Linien verlängert werden. Gültige Angaben siehe 'Skaleneinteilung'.

Gitter: Wenn diese Option aktiviert ist, werden die Y-Linien des Gitternetzes im Koordinatensystem eingeblendet. Die Skaleneinteilungen werden zu diesem Zweck mit gestrichelten Linien verlängert. Die Farbe der Gitternetzlinien kann von der Farbauswahlliste gewählt werden.

Beschreibung: Achsenbeschriftung: Wenn diese Option aktiviert ist, wird der Text, der im Eingabefeld angegeben wird, am oberen Ende der Y-Achse angezeigt. Für die X-Achse kann hier keine Beschriftung angegeben werden.

Anzeigeeinstellungen

Sie finden den Befehl **Anzeigeeinstellungen** im Kontextmenü des Trace-Baumes im rechten Teil des Hauptfensters des Trace-Editors. Er öffnet den Dialog **Anzeigeeinstellungen bearbeiten** im Trace-Editor, wenn der Name des Traces an oberster Stelle des Trace-Baumes selektiert ist. Ist eine darunter liegende Variable selektiert, öffnet der Dialog Darstellung Y-Achse.

Die Dialoge stehen auch zur Verfügung, wenn man sich im Konfigurationsdialog des Traces befindet.

In beiden Dialogen wird definiert, wie die Trace-Daten im Trace-Diagramm angezeigt werden. Das heißt, dass die Darstellung des Koordinatensystems wie Achsenkonfiguration, Farben, Gitternetz, Schrift etc. für die Anzeige der aufgezeichneten Werte der Variablen eingestellt wird. Die Einstellungen aus dem Dialog **Anzeigeeinstellungen bearbeiten** werden im standardmäßig verwendeten Einkanal-Modus verwendet. Die Einstellungen im Dialog **Darstellung Y-Achse** werden im Multikanal-Modus sichtbar.

Trace-Editor im Online-Modus

Die "Trace"-Funktionalität läuft wie eine Applikation auf dem Zielsystem. Eine aktivierte Trace-Aufzeichnung wird angezeigt, wenn das Trace-Objekt im Gerätebaum grün hinterlegt ist.

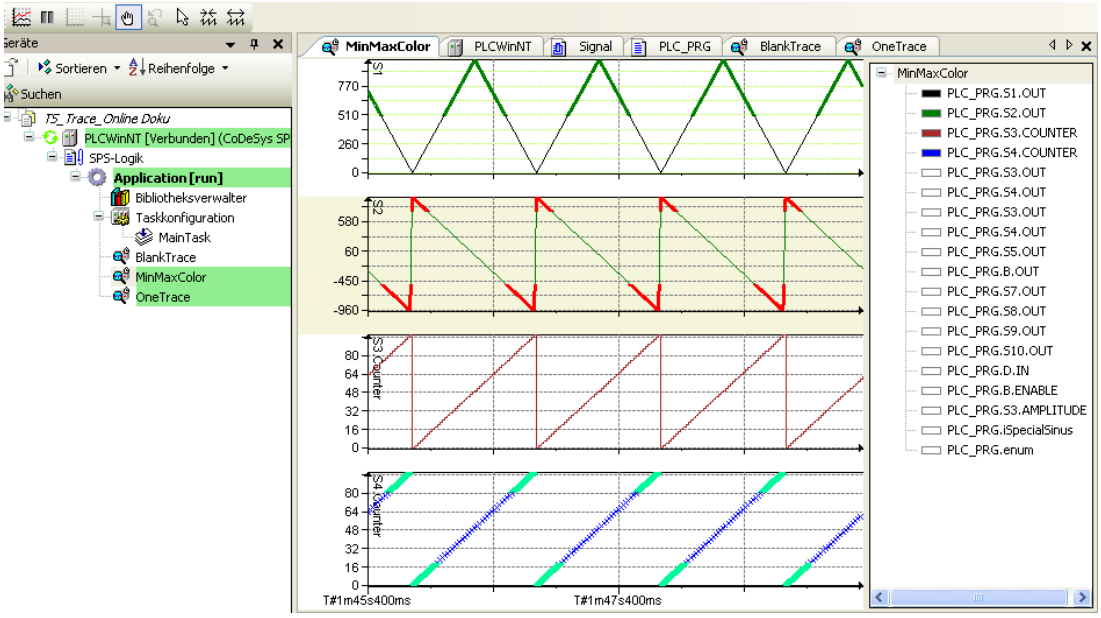


Abb. 425 Aktiver "Trace" im Gerätebaum

Eingeloggter Trace

Bevor eine konfigurierte Trace-Aufzeichnung auf der Steuerung für die dort laufende Applikation aktiv werden kann, muss die "Trace-Applikation" **explizit dorthin geladen werden** (Befehl 'Trace einloggen'). Wenn in der Applikation Änderungen vorgenommen wurden, wird die Trace-Aktivität automatisch beendet und die Konfiguration muss erneut geladen werden. Wenn aus der Applikation auf die Steuerung ausgeloggt und wieder eingeloggt wird, ohne dass Änderungen vorgenommen wurden, läuft die Trace-Aktivität auch ohne erneutes **Einloggen** weiter.

Konfiguration des Trace-Graphen

Die Dialoge für die Konfiguration sind auch im Online-Modus verfügbar und Änderungen können online vorgenommen werden. Der Dialog öffnet mit einem Doppelklick auf die Namen im Konfigurationsbaum im rechten Teil des Trace-Editors und das Kontextmenü dort enthält das Konfigurationskommando.

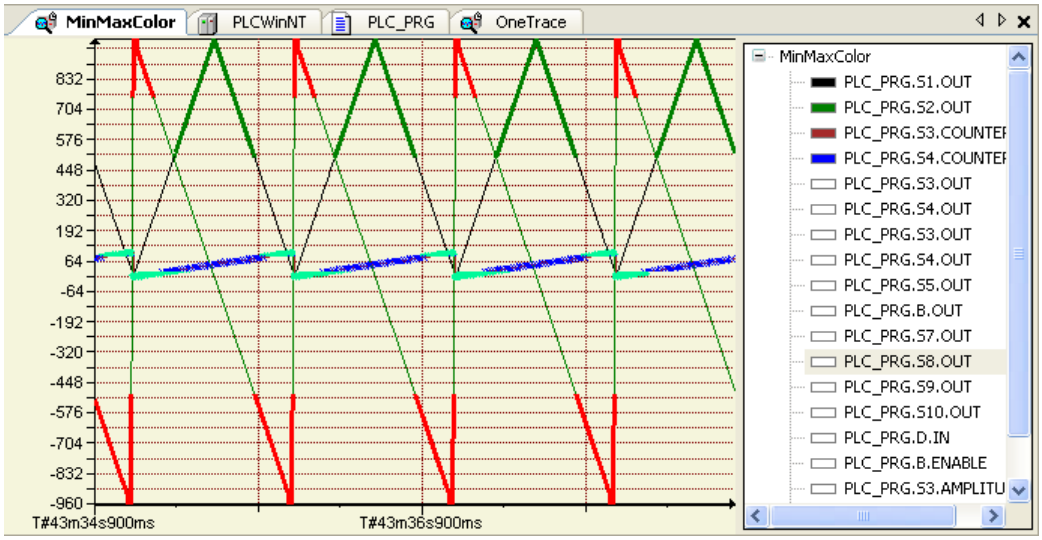


Abb. 426 Trace-Fenster im Online-Modus, Beispiel im Einzelkanal-Modus

Steuerung der Trace-Anzeige

Standardmäßig sind im Trace-Menü Befehle verfügbar, mit denen der aktuell im Trace-Fenster angezeigte Trace-Graph gesteuert werden kann (**Start/Stop Trace**, **Trigger zurücksetzen**, ...) bzw. der gerade angezeigte Ausschnitt aus den gesammelten Werten verändert werden kann (**Scrollen**, **Zoomen**, etc.).

Der Wertebereich des Trace-Graphen hängt von den spezifischen Einstellungen des Traces ab. Diese können jedenfalls durch Scrollen und Zoomen mittels Trace-Menü-Kommandos auch manuell manipuliert werden. Dies kann auch mittels Tastenkombinationen erfolgen. Beim vertikalen Scrollen der aktuellen Ansicht des Traces entlang der Y-Achse kann die **Nach-Oben-Pfeiltaste** für einen Bewegung nach oben und die **Nach-unten-Pfeiltaste** für eine Bewegung nach unten verwendet werden. Horizontales Scrollen entlang der X-Achse erfolgt mit Verwenden der **Nach-links-** oder **Nach-rechts-Pfeiltaste** bei gleichzeitig gedrückter **Alt-Taste**.

Ist die Multikanal-Option aktiviert, dann wird jeder Graph in seinem eigenem Diagramm angezeigt und ein selektierter Graph kann als einziger allein gescrollt werden mit den unten angegebenen Tastenkürzel. Um alle Graphen simultan zu scrollen, sind die Pfeiltasten mit zusätzlich gedrückter Strg-Taste zu verwenden, um auf der Y-Achse zu verschieben.

Alle Trace-Tastenkürzel sind in Tastenkombination für Trace aufgelistet.

Beachten Sie in diesem Zusammenhang die Zoom-Funktionalität der Scroll-Maus.

Befindet sich der Cursor im Trace-Fenster, wird der jeweilige Wert der X-Achse in der Statuszeile angezeigt.

Kindapplikation Trace

Existiert die zusätzliche Komponente 'Trace Manager' in Laufzeitsystem, dann wird die Trace-Konfiguration in einer eigenen Kindapplikation hinuntergeladen. Diese ist dann im Applikationsdialog des LZS im Geräteeditors aufgelistet und es ist möglich die Kindapplikation dort zu scannen und zu löschen.

Syntax einer Trace-Kindapplikation

<Applikationsname>.__<Trace-Name>

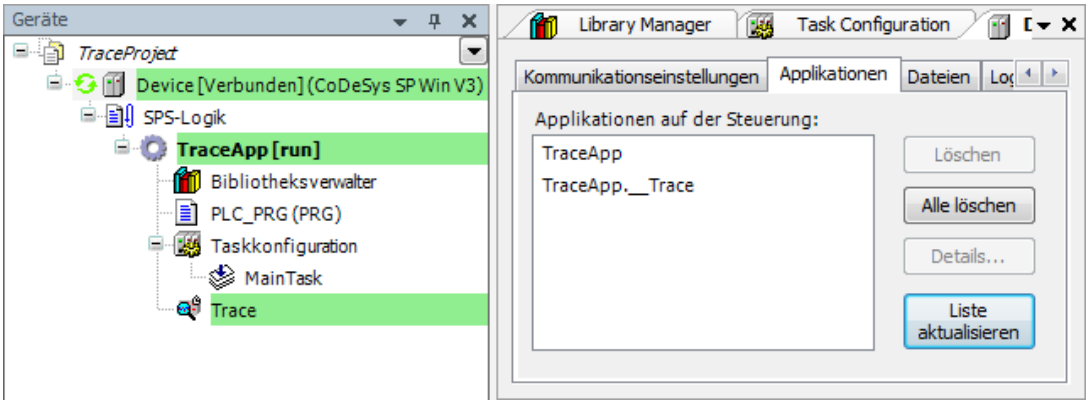


Abb. 427 Im Geräteeditor, Reiter Applikation: Trace-Applikation

13.18 Visualisierungseditor

Der Visualisierungs-Editor für das Erstellen von Visualisierungsobjekten wird vom Plug-In 'Visualization Editor' bereitgestellt. Sehen Sie die Beschreibung zu "Visualisierung" für einen Überblick und die Voraussetzungen für Visualisierung in »PLC Designer« V3.x.

Der Visualisierungs-Editor wird geöffnet, wenn auf ein Visualisierungsobjekt, das im POUs- oder Gerätefenster eingefügt wurde, ein Doppelklick oder der Befehl 'Objekt öffnen' angewendet wird.

Das Editorfenster ist betitelt mit " <Visualisierungsname>[<Objektpfad>].

Es gibt die folgenden Werkzeuge und Editoren, die zusammen mit dem Visualisierungs-Editor zur Erstellung einer Visualisierung verwendet werden:

1. Die Werkzeugbox, die die verfügbaren Visualisierungselemente zum Einfügen im Visualisierungs-Editor bereitstellt.
2. Der Eigenschaften-Editor, in dem die Attribute des gerade im Visualisierungs-Editor markierten Elementes angezeigt und bearbeitet werden.

Wenn mindestens einer der folgenden Editoren über den entsprechenden Befehl (standardmäßig im Menü Visualisierung) geöffnet wurde, wird der Visualisierungs-Editor zweigeteilt und enthält im oberen Teil die entsprechenden Registerblätter:

1. Der Schnittstellen Editor zum Definieren von Platzhaltern, wenn die Visualisierung in einer anderen Visualisierung referenziert (eingefügt) werden soll.
2. Der Tastaturbedienungs-Editor für das Verknüpfen von Aktionen mit Tasten bzw. Tastenkombinationen. Dabei ist zu beachten, dass das Gerät, auf dem die Visualisierung läuft, diese Tasten unterstützen muss.
3. Der Elementlisten-Editor, der eine Auflistung aller Elemente der aktuellen Visualisierung enthält, in dem Elemente selektiert und gelöscht werden können und ihre Position zwischen Vorder- und Hintergrund verschoben werden kann.

Bitte beachten: Für das Arbeiten mit Visualisierungen im Programmiersystem gelten die Voreinstellungen im Optionen-Dialog, Kategorie Visualisierung. Beispielsweise kann im Editorfenster ein Raster hinterlegt werden.

Hinweis: In Visualisierungen, die unter einem Gerät hängen, können geräteabhängige Einschränkungen bezüglich der Konfigurationsmöglichkeiten gelten (z.B.: Liste der verfügbaren Elemente, Schriftarten, Farben, Bildformate, max. Anzahl von Visualisierungselementen innerhalb einer Visualisierung, max. Anzahl von Visualisierungen pro Geräts).

Nach dem Einfügen in den Visualisierungs-Editor kann ein Element selektiert und seine Position und Größe direkt über "drag&drop" Mausaktionen verändert werden, bzw. seine Ausrichtung und Reihenfolge (Hintergrund, Vordergrund..) mit Hilfe der entsprechenden Visualisierungsbefehle definiert werden.

Weitere Konfigurationsmöglichkeiten eines Elements hängen vom Elementtyp ab. Alle Elementeigenschaften werden jeweils im Eigenschaften-Fenster angezeigt und können dort bearbeitet werden.

13.18.1 Visualisierungselemente einfügen, Werkzeugbox

Das Fenster **Werkzeuge** (Toolbox), das zusammen mit dem Visualisierungs-Editor verwendet wird, stellt Visualisierungselemente zum Einfügen ins Editorfenster bereit. Die Elemente werden Bibliotheken im Projekt bereitgestellt, die Auswahl im Werkzeuge-Fenster hängt vom gerade aktiven Visualisierungsprofil und im Fall von gerätezugehörigen Visualisierungen von der Gerätebeschreibung ab.

Das Werkzeuge-Fenster wird standardmäßig beim Öffnen des Visualisieruns-Editors mit geöffnet. Explizit kann es über den Befehl 'Werkzeuge' geöffnet werden, der standardmäßig im Menü 'Ansicht' zu finden ist. Das Werkzeuge-Fenster enthält die Kategorien "Visualisierung", "Comlex Controls" und "Windows Controls", deren zugehörige Visualisierungselemente jeweils mit Symbol und Namen aufgelistet sind. Mittels **drag & drop,** also Anwählen und Ziehen (bei gedrückter Maustaste) können die Visualisierungselemente ins Editorfenster eingefügt werden.

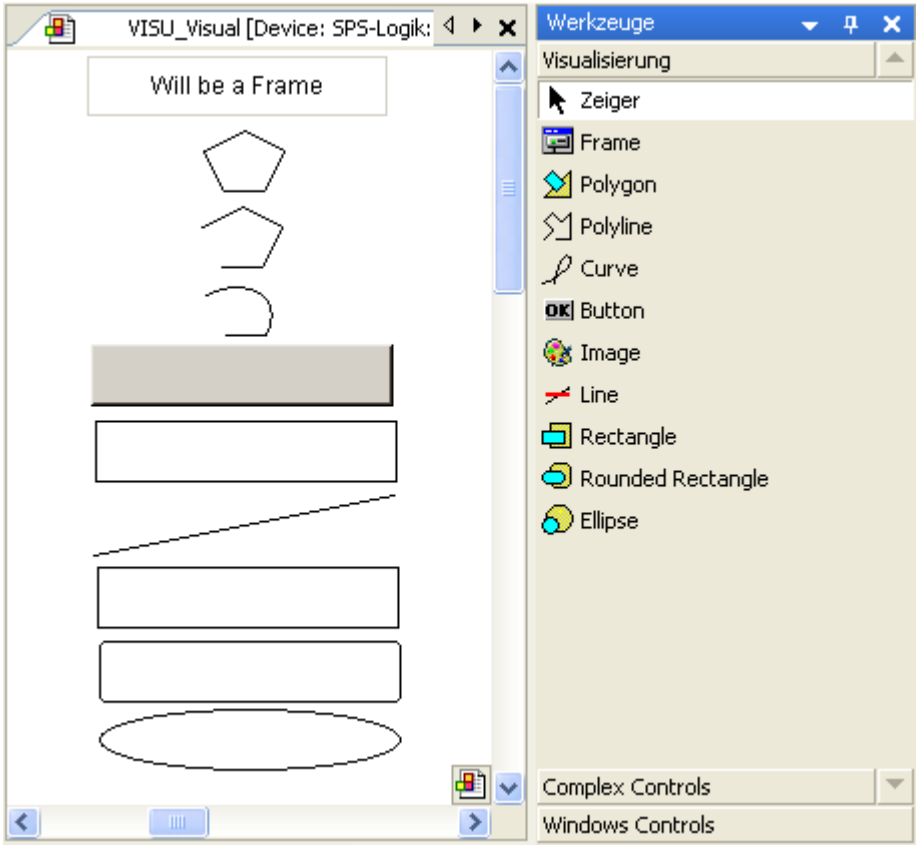


Abb. 428 Beispiel - Werkzeuge in Kategorie "Visual" und eingefügte Elemente

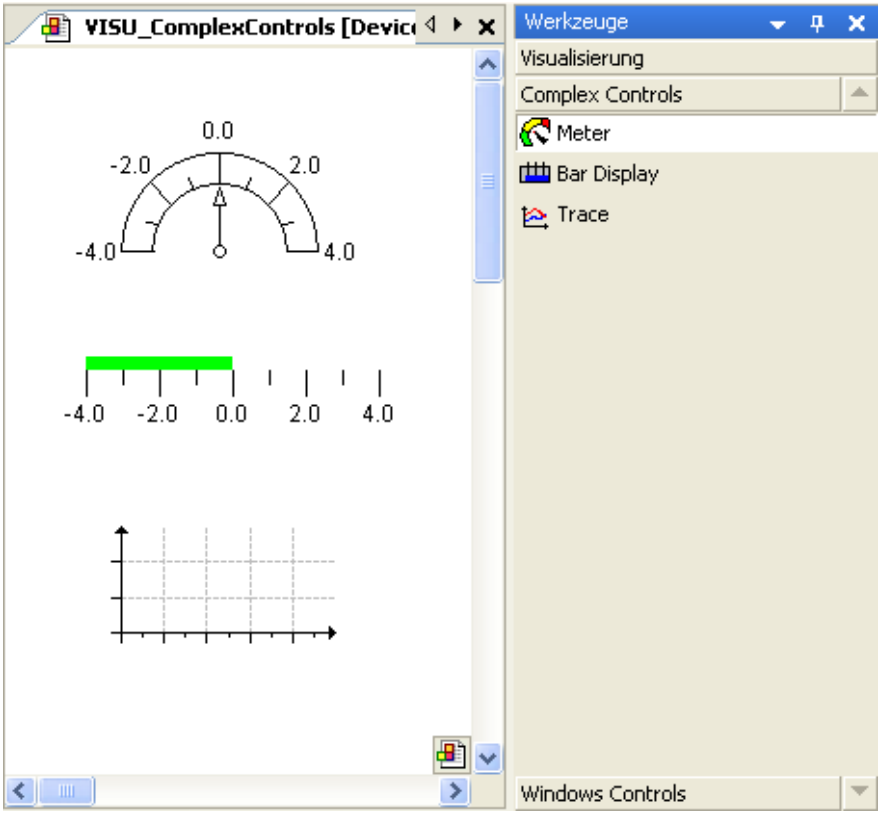


Abb. 429 Beispiel - Werkzeuge in Kategorie "Complex Controls" und eingefügte Elemente

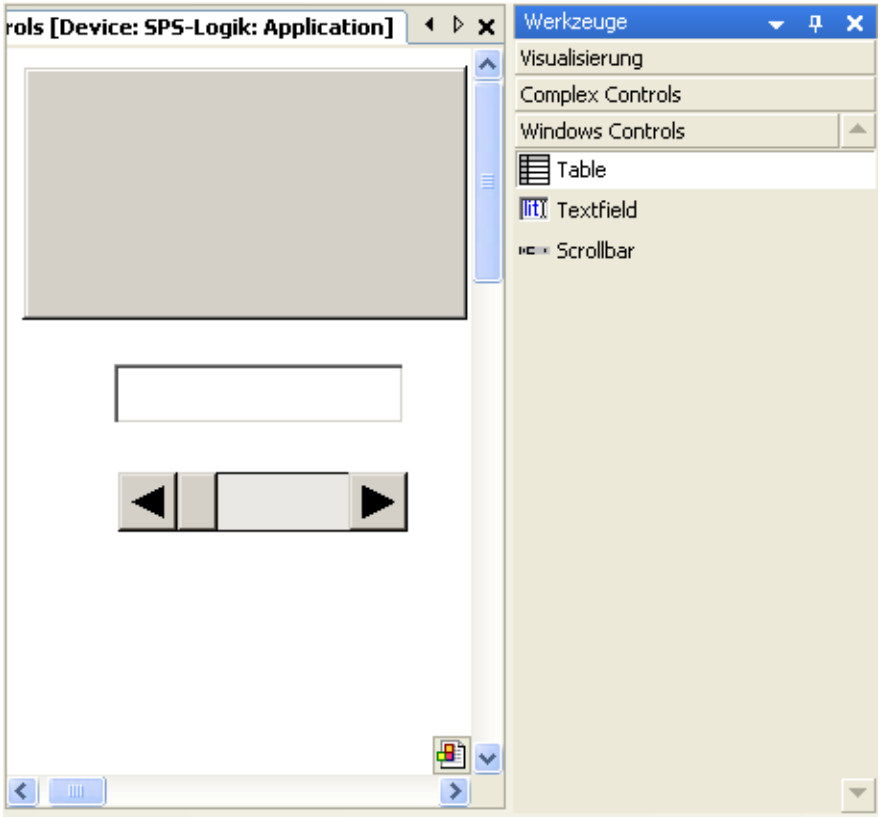


Abb. 430 Beispiel - Werkzeuge in Kategorie "Windows Controls" und eingefügte Elemente

Ein Pluszeichen am Cursor-Symbol zeigt an, dass ein Element gerade ins Editorfenster gezogen wird. Nach Loslassen der Maustaste wird das Element dort eingefügt.

Ein Element kann auch aus dem Untermenü Visualisierungselement einfügen eingefügt werden, das standardmäßig im Menü 'Visualisierung' zu finden ist. Hier steht die gleiche Auswahl an Elementen bereit wie im Werkzeuge-Fenster. Wenn nötig, kann das Visualisierungsmenü über den Anpassen-Dialog (Befehlskategorie 'Visualisierungskommandos') entsprechend umkonfiguriert werden.

Zu Einfügen anderer Visualisierungen in das gerade selektierte Frame-Element in der aktuellen Visualisierung kann über den Befehl Frame Auswahl , standardmäßig ebenfalls im Visualisierungsmenü verfügbar, ein entsprechender Auswahldialog geöffnet werden.

Nach dem Einfügen eines Elements kann es selektiert werden und Position, Größe, Anordnung und Ausrichtung direkt im Editor verändert werden, bzw. weitere Parameter im Eigenschaften-Fenster.

Die Standard-Bibliothek für Visualisierungslemente ist: VisuElems.library.

13.18.2 Elemente selektieren

Ein im Visualisierungseditor selektiertes Visualisierungselement zeigt kleine schwarze Quadrate in seiner Rahmenlinie, die mit einem weiteren Mausklick "angefaßt" und verschoben werden können, um Position oder Größe des Elements zu verändern.

Um ein Element auszuwählen (selektieren), ziehen Sie den Mauszeiger über das Element. Wenn das Cursor-Symbol eine Hand darstellt, drücken Sie die linke Maustaste.

Mehrfach-Selektion ist möglich. Dazu werden - jeweils bei gedrückter <Umschalt>-Taste - die einzelnen Elemente "angeklickt" oder mit gedrückter linker Maustaste ein Rechteck um die Elemente gezogen. Sehen Sie außerdem die Visualisierungskommandos 'Alles auswälen' und 'Alles deselektieren'.

Elemente können auch in der Elementliste selektiert werden. Dann können auch einzelne Unterelemente einer Gruppe ausgewählt werden.

Tastaturbedienung:

Wenn ein Element ausgewählt ist, kann die Selektion kann mit Hilfe der <Tabulator>-Taste auf das nächste Element in Einfügereihenfolge (Z-Reihenfolge) verschoben werden, mit <Umschalt> + <Tabulator> auf das entsprechend vorherige.

Mit der <Leertaste> wird in einem gerade selektierten Element ein Texteingabefeld geöffnet. Nach Eingabe einer Zeichenfolge wird diese mit der <Eingabe>-Taste für das Element (Elementeigenschaft "Text") übernommen.

Die Eigenschaften eines ausgewählten Elements werden unmittelbar im Eigenschaften-Fenster angezeigt und können dort bearbeitet werden. Sind mehrere Elemente ausgewählt, werden Änderungen an den Eigenschaften auf alle angewendet.

13.18.3 Position, Größe, Ausrichtung, Reihenfolge

Nach dem Einfügen eines Visualisierungselements können folgende Eigenschaften unmittelbar im Editorfenster verändert werden, wobei Größe und Position auch im Eigenschaften-Fenster bearbeitet werden können:

Position

Entweder Sie selektieren das Element mit einem Mausklick und ziehen es sofort, ohne die Maustaste loszulassen, an die gewünschte Position, oder Sie klicken erneut auf ein bereits selektiertes Element und bewegen es dann ebenfalls mit gedrückter Maustaste oder mit Hilfe der Pfeiltasten. Eine weitere Möglichkeit ist die "Position"-Werte in den Elementeigenschaften zu editieren.

Größe

Selektieren sie das Element und klicken dann mit der Maus auf eines der kleinen Quadrate in seiner Rahmenlinie. Das Cursor-Symbol - gekreuzte Doppelpfeile oder ein vertikaler oder horizontaler Doppelpfeil - zeigt jeweils an, in welche Richtung das Quadrat bewegt werden kann und mit ihm der entsprechende Teil der Rahmenlinie um die Größe des Elements zu verändern. Eine weitere Möglichkeit ist die "Position"-Werte in den Elementeigenschaften zu editieren.

Reihenfolge

Die Position eines Elements auf der Z-Achse einer Visualisierung, d.h. im Hintergrund oder Vordergrund bzw. irgendwo dazwischen, kann über die Befehle verändert werden, die standardmäßig im Untermenü Visualisierung/Reihenfolge Order verfügbar sind.

Ausrichtung

Um zwei oder mehrere Elemente rechts, links, oben, unten oder horizontal bzw. vertikal zentriert aneinander auszurichten, verwenden Sie bitte die entsprechenden Befehle im Menü Visualisierung / Reihenfolge.

Standardmäßig werden die Befehle zum Arbeiten im Visualisierungseditor (Befehlskategorie 'Visualisierungskommandos') im Menü 'Visualisierung' bereitgestellt. Wenn erforderlich, kann die Menüstruktur über den Anpassen-Dialog verändert werden.

13.18.4 Gruppierung von Elementen

Mehrere Elemente können ausgewählt und zu einer Gruppe zusammengefaßt werden. Sehen Sie hierzu die Beschreibung der Visualisierungsbefehle Gruppieren und Gruppierung aufheben. Unterlemente einer Gruppe könne nur über die Elementliste selektiert werden.

13.18.5 Visualisierungselemente - Eigenschaften

Wenn der Visualisierungs-Editor geöffnet ist, können Visualisierungselemente aus dem Werkzeuge-Fenster oder über das Visualisierungs-Menü eingefügt werden.

Nach dem Einfügen eines Elements in die aktuelle Visualisierung kann es selektiert werden und Position, Größe, Anordnung und Ausrichtung direkt im Editor (Mausaktionen bzw. Visualisieurngsbefehle) verändert werden. Die **Konfiguration** weiterer Parameter wie auch des Visualisierungsverhaltens erfolgt im Eigenschaften-Fenster.

Eigenschaften-Editor für Visualisierungselemente

Die Eigenschaften eines Visualisierungselementes - mit Ausnahme von Ausrichtung und Reihenfolge - können alle im 'Eigenschaften'-Editor konfiguriert werden. Dieser öffnet standardmäßig neben dem Visualisierungs-Editor oder wird explizit über den Befehl 'Eigenschaften' (standardmäßig im Ansicht-Menü) geöffnet.

Eine **Eigenschaft** kann durch Bearbeiten des Feldes **Wert** verändert werden. Dazu wird in diesem Feld abhängig vom Elementtyp ein Eingabefeld, eine Auswahlliste oder ein Dialog bereitgestellt, zu öffnen durch einen Doppelklick oder über die <Leertaste>, wenn das Feld bereits angewählt ist.

Bei Eigenschaften der Kategorie "Eingaben" / OnDialogClosed oder Eingaben / OnMouse-Aktionen wird die aktuelle Konfiguration in einer oder mehreren Zeilen unterhalb des "Konfigurieren..." Feldes. Zum Bearbeiten einer einzelnen Subkonfiguration klicken Sie in die entsprechende Zeile, um einen vereinfachten Konfigurationsdialog zu erhalten. Um die komplette Konfiguration für die Maus-Aktion zu ändern, klicken Sie ins Feld "Konfigurieren..." um den ausführlichen Standardkonfigurationsdialog zu erhalten.

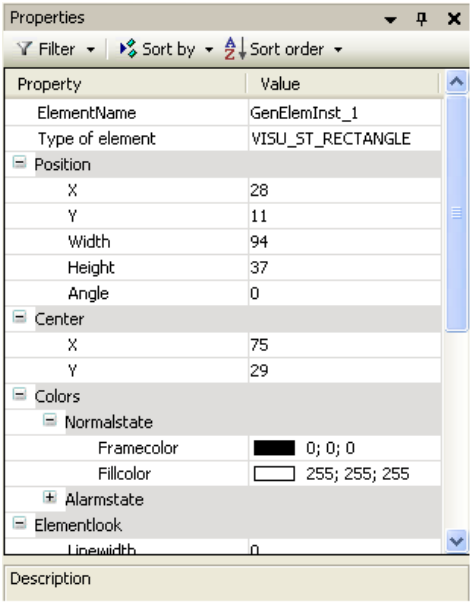


Abb. 431 Eigenschaften-Editor, Beispiel

Das Arbeiten in der Liste der Eigenschaften kann mit Hilfe von Standard- **Sortier- und Filterfunktionen** erleichtert werden.

Hinweis: In Visualisierungen, die einem Gerät zugewiesen sind, können geräteabhänige Einschränkungen der Konfigurationsmöglichkeiten gelten (beispielsweise bzgl.: Liste der verfügbaren Elemente, Schriftarten, Farben, Bildformate, max. Anzahl von Elementen in einer Visualisierung, max. Anzahl von Visualisierungen unterhalb des Geräts).

Die folgenden Tabellen enthalten Informationen zu allen möglichen Elementeigenschaften. Sie können die Anzeige für die einzelnen Kategorien jeweils durch Mausklick auf den Kategorientitel explizit öffnen oder schließen.

+Columns (nur für Table-Element)

Das Table-Element visualisiert eine Variable vom Array- oder Strukturtyp. Der Index des Arrays bzw. die Komponente der Struktur wird in einer Spalte bzw. Zeile dargestellt. Bei zweidimensionalen Arrays bzw. einem Array von Strukturen in mehreren Spalten.

Hier wird das Aussehen der Tabellenspalten definiert, in welchen die Werte der einzelnen Arrayelemente angezeigt werden.

Für jede zu einem bestimmten Index oder einer einzelnen Komponente gehörenden Spalte ist eine eigene Konfiguration möglich.

Column		
[index]	Spaltenindex des Table Elements; z. B. "[0]" für die einzige Spalte, die die Elemente anzeigt von array[2..8] of INT; z.B. "[0]" , "[1]" oder "[2]" für die drei Spalten, die die Elemente von array[1..3][1..10] of INT anzeigen.	Ta
Column header	standardmäßig wird der Name des Array/der Struktur mit dem zur Spalte gehörenden Index/Komponente als Überschrift genommen; die Spaltenbeschriftung kann geändert werden indem hier ein neuer Titel eingetragen wird	Ta
Column width	Breite der Spalte	Ta
Text in headline	Ausrichtung der Spaltenbeschriftung: LEFT (links) CENTER (zentriert) RIGHT (rechts)	Ta
Use template	Bei Aktivierung des Kontrollkästchens wird in jede Zelle dieser Tabellenspalte ein weiteres Visualisierungselement (vom Typ Rectangle, Rounded Rectangle oder Circle) eingefügt. Die Eigenschaftenliste wird automatisch um die Eigenschaften dieses Elements erweitert, das wie üblich konfiguriert werden kann.	Ta

Column		
Use text alignment in headline	Die Aktivierung dieses Kontrollkästchens hat nur bei gleichzeitiger Aktivierung des Kontrollkästchens "Use template" einen Effekt: bei Aktivierung werden die im eingefügten Template getroffenen Einstellungen für Schrift(größe) und Ausrichtung auch auf die Spaltenüberschrift angewendet.	Ta
Row header	Ein- bzw. Ausschalten der Anzeige der Zeilenbeschriftung (in Form von Zeilenindices des dargestellten Arrays)	Ta Ta
Column header	Ein- bzw. Ausschalten der Anzeige der (unter "Column heade"r) definierten Spaltenüberschriften	
Row height	Höhe der Zeilen in Tabelle	Ta
Row header width	Breite der Spalte mit Zeilenbeschriftungen	Ta
Scrollbar size	Breite des Scrollbars	Ta

Bild-ID, Rahmen anzeigen, Abschneiden,Transparent, Transparent color, Skalierungstyp, Trace

Static ID	Kennzeichner der Bilddatei, die für dieses Element verwendet werden soll (statische Definition) Geben Sie die ID der Bilddatei ein, wie sie in der entsprechenden Bildersammlung definiert ist (STRING); der Name der Sammlung sollte dabei vorangestellt werden, um den Eintrag eindeutig zu machen (dies ist nicht nötig, wenn die Bilddatei im GlobalImagePool verwaltet wird, da diese Sammlung immer als erste durchsucht wird). Beispiel: "imagepool2.button_image".	Im
Show frame	Wenn diese Option aktiviert ist, wird die Bilddatei mit einem Rahmen dargestellt.	F,Im
Clipping	Wenn diese Option zusammen mit der "Scaletype"-Einstellung "Fixed" (s.u.) aktiviert ist, wird im Falle eines Frame-Elements nur der Teil der Visualisierung, der in den Frame passt, dargestellt.	F,Im
Transparent	Wenn aktiviert, wird die Farbe, die in Eigenschaft 'Transparent color' (s.u.) angegeben ist, transparent dargestellt.	Im
Transparent color	Die Schaltfläche öffnet den Farbauswahldialog zur Auswahl einer Farbe, die transparent dargestellt werden soll, wenn Option "Transparent" (s.o.) aktiviert ist.	Im

Scaletype	Definition, wie die Bilddatei auf Änderungen der Elementrahmengröße reagieren soll. ISOTROPIC: Das Bild behält seine Proportionen; d.h. das Verhältnis Höhe-Breite bleibt erhalten, auch wenn Höhe und Breite des Elementrahmens separat verändert werden. ANISOTROPIC: Das Bild paßt sich der Größe des Elementrahmens an; d.h. Höhe und Breite können unabhängig voneinander verändert werden. FIXED: Das Bild behält die Originalgröße, auch wenn sich die Größe des Elementrahmens ändert. Beachten Sie, ob auch Option 'Clipping' aktiviert ist! (s.o)	F,Im
Trace	Name des Trace, der als Eingang für das Visualisierungselement Trace dient. Anzugeben ist der vollständige Name des Trace, d.h. ergänzt mit dem Pfad des Trace innerhalb des Garätebaums, z.B. "PLCWinNT.Application.Trace".	Tr

Static ID	Kennzeichner der Bilddatei, die für dieses Element verwendet werden soll (statische Definition) Geben Sie die ID der Bilddatei ein, wie sie in der entsprechenden Bildersammlung definiert ist (STRING); der Name der Sammlung sollte dabei vorangestellt werden, um den Eintrag eindeutig zu machen (dies ist nicht nötig, wenn die Bilddatei im GlobalImagePool verwaltet wird, da diese Sammlung immer als erste durchsucht wird). Beispiel: "imagepool2.button_image".	R	rR	E		L		pL		F	Bu	Im

Show frame	Wenn aktiviert: Die Bilddatei wird mit einem Rahmen dargestellt.	R	rR	E		L		pL		F	Bu	Im
Clipping	Wenn diese Option zusammen mit der "Scaletype"-Einstellung "Fixed" (s.u.) aktiviert ist, wird im Falle eines Frame-Elements nur der Teil der Visualisierung, der in den Frame paßt, dargestellt.	R	rR	E		L		pL		F	Bu	Im

Transparent	Wenn aktiviert, wird die Farbe, die in Eigenschaft 'Transparent color' (s.u.) angegeben ist, transparent dargestellt.	R	rR	E		L		pL		F	Bu	Im
Transparent color	Schaltfläche öffnet den Farbauswahldialog zur Auswahl einer Farbe, die transparent dargestellt werden soll, wenn Option "Transparent" (s.o.) aktiviert ist.	R	rR	E		L		pL		F	Bu	Im

Scaletype	Definition, wie die Bilddatei auf Änderungen der Elementrahmengröße reagieren soll. ISOTROPIC: Das Bild behält seine Proportionen; d.h. das Verhältnis Höhe-Breite bleibt erhalten, auch wenn Höhe und Breite des Elementrahmens separat verändert werden. ANISOTROPIC: Das Bild paßt sich der Größe des Elementrahmens an; d.h. Höhe und Breite können unabhängig voneinander verändert werden. FIXED: Das Bild behält die Originalgröße, auch wenn sich die Größe des Elementrahmens ändert. Beachten Sie, ob auch Option 'Clipping' aktiviert ist! (s.o)											Im

Position Position, Breite, Höhe, ShowCursor, Positionspunkte (X, Y, Width, Height, points)

Hinweis: x/y = 0/0 bezeichnet die obere linke Ecke des Editorfensters, x-Achse= horizontale Achse, y-Achse=vertikale Achse.

X	Horizontale Position in Pixel. 0=linker Fensterrand	R,rR.E.L,F,Bu,Im M,BD,Tr Ta,Te,Sc
Y	Vertikale Position in Pixel. 0=oberer Fensterrand	R,rR.E.L,F,Bu,Im M,BD,Tr Ta,Te,Sc
Width	Breite des Elements in Pixel	R,rR.E.L,F,Bu,Im M,BD,Tr Ta,Te,Sc
Height	Höhe des Elements in Pixel	R,rR.E.L,F,Bu,Im M,BD,Tr Ta,Te,Sc
ShowCursor	Bei Aktivierung des Kontrollkästchens werden die Werte der überwachten Variablen(Trace-Variablen) an der Position des Mauszeigers angezeigt.	Tr
Points		
[0], X / Y ..... [n], X / Y	X/Y-Positionen (pixel) der einzelnen Elementpunkte. [0] ist die Position des Startpunkte, die folgenden Punkte werden fortlaufend nummeriert.	P,pL,C

Center Zentrum (X, Y)

X	X-Position des Drehpunktes (center, ⊕) des Elements	R,rR,E,P,L,pL,C,F,Bu,Im
Y	Y-Position des Drehpunktes (center, ⊕) des Elements	R,rR,E,P,L,pL,C,F,Bu,Im

Colors Farben (Color, Bar color, Bar background, Framecolor, Alarm value, Condition, AlarmColor, Use color areas, Durable color areas, Begin of area, End of area, Color, Framecolor, Fillcolor, Normalstate, Alarmstate)

- - Wählen sie eine Farbe von der Auswahlliste oder verwenden Sie den Farbauswahldialog (Schaltfläche ...)

Color	Statische Definition: Farbe des Elements im Normalzustand	L,F,Bu,Im, Sc,
Bar color	Farbe des Balkens	BD
Bar background	Die Hintergrundfarbe des Balkens ist standardmäßig weiß; bei Aktivierung des Kontrollkästchens wird die Hintergrundfarbe in schwarz geändert	BD
Framecolor	Farbe des Rahmens, der die Balkenanzeige umgibt, sofern das Kontrollkästchen "Element frame" aktiviert wurde	BD
Alarmcolor		
Alarm value	Schranke, die Alarmzustand angibt	BD
Condition	Der Alarmzustand wird ausgelöst, wenn der aktuelle Wert der Eingabevariablen den "Alarm value" überschreitet (GREATER_THAN) bzw. unterschreitet (LESS_THAN)	BD
AlarmColor	Statische Definition: Farbe des Elements im Alarmzustand, d.h. wenn die entsprechende Variable TRUE ist	L,F,Bu,Im, BD, Sc
Use color areas	Bei Aktivierung dieses Kontrollkästchens werden die angelegten Farbbereiche angezeigt	BD
Color areas	Anklicken der Taste Create new verursacht das Anlegen eines neuen Farbbereichs, dessen Eigenschaften in den unterhalb aufgehenden Feldern editiert werden können	
Durable color areas	Standardmäßig wird nur der Farbbereich angezeigt, der den aktuellen Wert der Eingabevariablen enthält. Nach Aktivierung des Kontrollkästchens werden alle angelegten Farbbereiche angezeigt.	M
Areas	Dieser Ordner enthält alle angelegten Farbbereiche.	
[Number]	Aufsteigende Nummerierung des Farbbereichs, kann nicht editiert werden. Nach Anklicken von Delete wird der zugehörige Farbbereich gelöscht und die Nummerierung neu vergeben.	

Begin of area	Wert, der den Farbbereich von unten begrenzt	M,BD
End of area	Wert, der den Farbbereich von oben begrenzt	M,BD
Color	Farbe, in der der Farbbereich dargestellt werden soll	M,BD
Normalstate		
Framecolor	Statische Definition: Farbe des Elements im Normalzustand	R, rR, E
Fillcolor	Statische Definition: Füllfarbe des Elements im Normalzustand	R, rR, E
Alarmstate		
Framecolor	Statische Definition: Farbe des Elementrahmens im Alarmzustand	R, rR, E
Fillcolor	Statische Definition: Füllfarbe des Elements im Alarmzustand	R, rR, E

Elementlook Aussehen, Linienstärke, Linientyp, Neigung, Füllattribute, Rahmenattribute (Linewidth, Linestyle, FromTopLeft (declination), Fillattributes, Frameattributes)

Dies sind statische Definitionen bezüglich der Rahmenlinie und der Füllung des Elements.

Zur dynamischen Definition dieser Eigenschaften siehe Kategorie 'Look variables'.

Linewidth	Dicke der Rahmenlinie in Pixels; "0" entspricht "1"	R,rR,E,P,L,pL,C,F,Im, Te
Linestyle	Linientyp: PS_SOLID ____ PS_DASH _ _ _ PS_DOT . . . PS_DASHDOT _ . _ . PS_DASHDOTDOT _ . . _ . . PS_HOLLOW (unsichtbar)	L
FromTopLeft	Wenn diese Option aktiviert ist, fällt die Linie von links oben nach rechts unten, ansonsten steigt sie von links unten nach rechts oben an.	L
Fillattributes	Sichtbarkeit der Rahmenlinie: BS_SOLID: sichtbar BS_HOLLOW: unsichtbar	R,rR,E,P,pL,C, Te

Frameattributes	Typ der Rahmenlinie: PS_SOLID ____ PS_DASH _ _ _ PS_DOT . . . PS_DASHDOT _ . _ . PS_DASHDOTDOT _ . . _ . . PS_HOLLOW (unsichtbar)	R,rR,E,P,pL,C,F,Im, Te
m_ShadowType	Typ der Schattierung der Außenlinie: DEEPEND (Element erscheint vertieft) NONE (ohne) RAISED (Element erscheint erhöht)	Te

Buttonheight Schaltfläche (nur für Button)

Buttonheight	Bestimmt die Relief-Ansicht des Schaltflächen-Elements; "Höhe" in Pixel ( )	Bu

Bitmapinfo Bilddatei (nur für Button: Static ID, Scaletype)

Static ID	siehe oben	Bu
Scaletype	siehe oben	Bu

Texts Texte, Tooltip (Text, Tooltip)

Statische Definition des Element-Textes. Zur dynamischen Definition siehe Kategorien 'Text variables' und 'Dynamic texts'.

Zur Definition von Schriftart und Ausrichtung siehe Kategorie 'Text properties' (statisch) und 'Font variables' (dynamisch).

Text	Text, der im Element erscheinen soll. Im Falle des Elements "Tabelle" wird der Text im Vordergrund quer über die ganze Tabelle geschrieben. Beispiel: "Ein/Aus" siehe auch bei Kategorien 'Text variables' and 'Dynamic texts' Beispiel: "On/Off" siehe auch: Kategorien "Text variables" und "Dynamic texts" Hinweis: Text kann auch "inline" eingegeben werden. Wenn das Element im Visualisierungseditor selektiert ist, kann durch Drücken der <Leertaste> ein Eingabefeld geöffnet werden.	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te
Tooltip	Text, der im Online-Modus alsTooltip erscheinen soll, wenn der Cursor auf einem Element plaziert ist. when the cursor is placed on the element. Example: "on/off button of machine 1" Beispiel: "On/Off"-Button mit Tooltip "Ein/aus Maschine 1"	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc

Text properties Texteigenschaften, Font (HorizontalAlignment, VerticalAlignment, Font)

Dies sind statische Definitionen, für eine dynamische Definition der Schrift siehe Kategorie 'Font variables'.

HorizontalAlignment	Horizontale Ausrichtung des Elementtextes: LEFT / HCENTER / RIGHT	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc
VerticalAlignment	Vertikale Ausrichtung des Elementtextes: TOP / VCENTER / BOTTOM	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc
Font	Font-Typ,- Farbe und -Größe des Elementtextes; über Schaltfläche ... kann der Standarddialog zur Font-Auswahl geöffnet werden	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc

Arrow (only for Meter: Arrowtype, Arrowcolor, Arrow start, Arrow end, Additional arrow)

Arrowtype	Das Aufklappmenü bietet vier verschiedene Formen für den Pfeil an:dünne Nadel (THIN_NEEDLE), dünner Pfeil (THIN_ARROW),normale Pfeil (NORMAL_ARROW), WIDE_ARROW (breiter Pfeil)	M
Arrowcolor	Farbe des Pfeils	M
Arrow start	Winkel (in Grad) zwischen linkem Rand des Tachometers und der horizontalen Achse	M
Arrow end	Winkel (in Grad) zwischen rechtem Rand des Tachometers und der horizontalen Achse	M
Additional arrow	Bei Aktivierung dieses Kontrollkästchens wird ein zweiter Pfeil innerhalb des Skalierung angezeigt, der dem ersten Pfeil entgegen gerichtet ist	M

Bar (Diagram type, Orientation, Running direction)

Diagram type	Das Aufklappmenü bietet drei Möglichkeiten zur Positionierung des Balken in Bezug auf die Skalierung: Balken lieget innerhalb der Skalierung (BAR_INSIDE_SCALE), Skaliereung liegt innerhalb des Balkens (SCALE_INSIDE_BAR),Skalierung ist neben dem Balken (SCALE_BESIDE_BAR)	BD
Orientation	Wählen Sie zwischen horizontaler (HORIZONTAL) und vertikaler (VERTICAL) Anordnung des Balken. Im Fall des Scrollbars resultiert die Orientierung (HORIZONTAL oder VERTICAL) aus dem Verhältnis von Länge zu Breite des Scrollbars und kann nicht editiert werden.	BD SC
Running direction	Wählen Sie aus den Optionen LEFT_RIGHT (für eine von links nach rechts wachsenden Balken) und RIGHT_LEFT im Fall eines horizontalen Balkens, bzw. unter BOTTOM_UP (für einen von unten nach oben wachsenden Balken) und TOP_DOWN im Fall eines vertikalen Balkens	BD SC

Scale (Scale start, Scale end, Main Scale, Sub Scale, Frame inside, Frame outside, Element Frame)

Scale start	Wert, der die Skalierungsanzeige von unten begrenzt.	M,BD
Scale end	Wert, der die Skalierungsanzeige von oben begrenzt.	M,BD
Main scale	Abstand zwischen zwei Strichen der Grobskala.	M,BD
Sub scale	Abstand zwischen zwei Strichen der Feinskala; kann auf 0 gesetzt werden, falls eine weiter Unterteilung der Grobskala nicht gewünscht ist.	M,BD
Frame inside	Dieses Kontrollkästchen ist standardmäßig aktiviert; bei Deaktivierung wird der untere Rand der gebogenen Skalierungsanzeige ausgeblendet.	M
Frame outside	Dieses Kontrollkästchen ist standardmäßig aktiviert; bei Deaktivierung wird der obere Rand der gebogenen Skalierungsanzeige ausgeblendet.	M
Element Frame	Dieses Kontrollkästchen ist standardmäßig deaktiviert; bei Aktivierung wird eine Rahmen um das Balkenelement gezeichnet	BD

Label (Unit, Font, Scale Format)

LabelPosition	Wählen Sie "OUTSIDE" um die Skalierungsbeschriftung auf der Außenseite der Skala zu platzieren, "INSIDE" um sie auf die Innenseite zu setzen	M
Unit	String, der unterhalb des Pfeils (M) bzw. unterhalb des Mittelpunkts der Skalierung (BD) angezeigt wird (gedacht für die Eingabe der Skalierungseinheit)	M,BD
Font	Schrift, in der Einheit und Skalierungsbeschriftung dargestellt werden	M,BD
Scale Format (C-Syntax)	Benutzen Sie die C Syntax zur Angabe der Formatierung der Skalierungsbeschriftung; z.B. führt die Eingabe des Strings "%3.2f s" in diesem Feld zur Anzeige der Skalierungsbeschriftungen mit 3 Stellen, davon 2 Nachkommastellen, gefolgt vom Buchstaben "s"	M,BD
max. text width of labels	Wert, der die maximale Breite der Skalierungsbeschriftung vorgibt; in der Regel wird dieser Wert automatisch richtig festgesetzt. Benützen Sie diese Vorgabe nur, falls die automatische Anpassung nicht zum gewünschten Ergebnis führt.	M
text height of labels	Wert, der die Höhe der Skalierungsbeschriftung vorgibt; in der Regel wird dieser Wert automatisch richtig festgesetzt. Benützen Sie diese Vorgabe nur, falls die automatische Anpassung nicht zum gewünschten Ergebnis führt.	M

Bitmap ID variable Bild-ID (nur für Bild-Element (Image))

Bitmap ID	Projektvariable, die die ID der Bilddatei definiert (siehe Bildersammlung)	Im

Absolute movement Bewegung Movement, X, Y, Rotation, Scaling, Interior Rotation)

Absolute Bewegung: Das Element kann bewegt werden, indem die X- und Y-Position (Pixels) der linken oberen Ecke des Elements durch eine Integer-Variable verändert wird. Hier werden absolute Koordinatenwerte verwendet.

(Siehe im Gegensatz dazu:: 'Relative Bewegung' für das Bewegen bestimmter Ecken/Seitenlinien des Elements gegenüber einem festen Ursprung.)

Movement		
X	Die hier eingetragene Integer-Variable definiert die aktuelle X-Position der linken oberen Ecke des Elements in Pixels definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben	R,rR,E,P,L,pL,C,F,Bu,Im
Y	Die hier eingetragene Integer-Variable definiert die aktuelle Y-Position der linken oberen Ecke des Elements in Pixels definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben	R,rR,E,P,L,pL,C,F,Bu,Im
Rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um den Rotationspunkt rotiert; positive Werte=mathematisch positiv=Uhrzeigersinn. Hinweis: Das Element selbst dreht sich nicht, im Gegensatz zu 'Interior rotation' (s.u.) Der Rotationspunkt (Zentrum ⊕) wird bei einem Mausklick auf das Element sichtbar. Er kann gedrückt gehaltener Maustaste verschoben werden.	R,rR,E,P,L,pL,C,F,Bu,Im
Scaling	Die hier eingetragene Integer-Variable definiert den aktuellen Skalierungsfaktor (Prozent); die Elementgröße wird linear entsprechend dieses Wertes verändert. Implizit wird der Wert durch 1000 dividiert, so dass es nicht nötig ist, REAL-Variablen zu verwenden um das Element zu verkleinern. Die Skalierung bezieht sich immer auf den Rotationspunkt (Zentrum). Der Rotationspunkt (Zentrum ⊕) wird bei einem Mausklick auf das Element sichtbar. Er kann gedrückt gehaltener Maustaste verschoben werden.	R,rR,E,P,L,pL,C,F,Bu,Im
Interior rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um seinen Rotationspunkt gedreht wird; positive Werte=mathematisch positiv=Uhrzeigersinn. Im Gegensatz zu 'Rotation' (s.o.) rotiert das Element selbst. Der Rotationspunkt (Zentrum ⊕) wird bei einem Mausklick auf das Element sichtbar. Er kann gedrückt gehaltener Maustaste verschoben werden.	R,rR,E,P,L,pL,C,F,Bu,Im

Relative movement (Movement topleft, Movement bottomright, X, Y)

Die obere, linke, untere bzw. rechte Kante des Elements wird um den durch eine Integer-Variable gegebenen Wert (Pixels) in X- bzw. Y-Richtung verschoben. Im Gegensatz zur absouten Bewegung (s.o.) wird eine "relative Position definiert, d.h. der Abstand zur ursprünglichen Position. Das Element kann dadurch in der Form verändert werden.

Positive Werte verschieben die horizontalen Kanten nach unten bzw. die vertikalen nach rechts.

Siehe auch: 'Absolute movemen't für das Verschieben des ganzen Elements.

Movement topleft		
X	Anzahl der Pixel um die linke Kante in X-Richtung verschoben wird.	R,rR,E,L,F,Bu,Im
Y	Anzahl der Pixel um die obere Kante Ecke in Y-Richtung verschoben wird.	R,rR,E,L,F,Bu,Im

Movement bottomright		
X	Anzahl der Pixel um die rechte Kante in X-Richtung verschoben wird.	R,rR,E,L,F,Bu,Im
Y	Anzahl der Pixel um die untere Kante Ecke in Y-Richtung verschoben wird.	R,rR,E,L,F,Bu,Im

Dynamic values Dynamische Werte (für Elemente mit mehreren Positionspunkten: pPointArray, Count)

pPointArray	Die hier eingegebene Variable muss ein Zeiger auf ein Array der Struktur **VisuStructPoint** sein. VisuStructPoint beschreibt in seinen Komponenten **iX** und **iY** die X/Y-Koordinaten eines Punktes des Elements. Das Array beschreibt alle Punkte des Elements und die Struktur kann dynamisch aus dem Projekt gefüllt werden. Beachten Sie, dass die Anzahl der Element-Punkte explizit definiert sein muss (s.u.), weil die Verwendung eines Zeigers für die dynamische Definition der Punkte hier nicht erlaubt, die Anzahl der Punkte zu kontrollieren. Beispiel: pPoints : POINTER TO ARRAY[0..100] OF VisuStructPoint;	P,pL,C
Count	Hier muss eine Integer-Variable eingegeben werden, die die Anzahl der Punkte des Elements definiert. Beispiel: iCount : INT:=24; Dies heißt, dass das Element 24 einzelne Punkte hat. Diese Festlegung ist nötig, weil die Definition der einzelnen Punkte über einen Zeiger (s.o. pPointArray) erfolgt und dies nicht erlaubt, die Anzahl zu kontrollieren.	P,pL,C

Text variables Textvariablen (Text Variable, Tooltip Variable)

Dies sind dynamische Definitionen; siehe auch Kategorie 'Dynamic texts' für die Verwendung von Textlisten.
(Für statische Definitionen siehe Kategorie 'Texts'.)

Text Variable	String-Variable, die den Element-Text definiert. Im Fall einer Tabelle wird der Text im Vordergrund quer über die ganze Tabelle gelegt. Siehe auch die Kategorien "Text variables" und "Dynamic texts".	R,rR,E,P,L,pL,C,F,Bu,Im Te,Sc
Tooltip Variable	String-Variable die den Tooltip-Text für das Element definiert	R,rR,E,P,L,pL,C,F,Bu,Im Te,Sc

Dynamic texts Dynamische Texte, Textliste (TextList, TextIndex, TootipIndex)

Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung).

Als weitere Möglichkeit einer dynamischen Text-Definition kann der Text über eine String-Variable geliefert werden, siehe Kategorie 'Text variables'.
(Zu statischen Text-Definitionen siehe Kategorie 'Texts'.)

TextList	Name der Textliste (String) wie im POUs- oder Geräte-Fenster verwendet; z.B. " 'errorlist' "	R,rR,E,P,L,pL,C,F,Bu,Im Te,Sc
TextIndex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID (z.B. " 'M1' ") oder String-Variable (z.B. "error_22")	R,rR,E,P,L,pL,C,F,Bu,Im Te,Sc
TooltipIndex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID (z.B. " 'T1' ") oder String-Variable (z.B. "error_tooltip_22")	R,rR,E,P,L,pL,C,F,Bu,Im Te,Sc

Font variables Fontvariablen (Fontname, Height, Flags, CharSet, Color)

Dies sind dynamische Definitionen zum Font des Element-Textes über eine Projektvariable.

(Für statische Definitionen siehe Kategorie 'Texts'.)

Fontname	String-Variable, die den Namen des Fonts, der für den Element-Text verwendet werden soll, angibt; (Namensangabe wie im Standard-Font-Dialog); z.B. "plc_prg.font_var" (font_var:='Arial';)	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc
Height	Integer-Variable, die die Größe des Element-Textes in Pixel definiert; (Angabe wie im Standard-Font-Dialog); z.B. "prog1.font_height" (font_height:=16;)	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc
Flags	DWORD-Variable, die über die unten genannten Flag-Werte die Font-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird.: 1 italic 2 bold 4 underlined 8 canceled z.B. "prog2.font_type" (wenn font_type:=6, wird der Text fett und unterstrichen dargestellt)	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc
CharSet	Der Zeichensatz, der für den Font verwendet werden soll, kann über die Standard-Zeichensatz-Nummer definiert werden. Mit einer DWORD-Variablen kann diese Nummer angegeben werden (siehe auch die "Skript"-Definition im Standard-Font-Dialog)	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc
Color	Über eine DWORD-Variable kann hier die Farbe des Element-Textes definiert werden.	R,rR,E,P,L,pL,C,F,Bu,Im Ta,Te,Sc

Color variables Farbvariablen (ToggleColor, Color, Alarmcolor, Normalstate, Alarmstate, Framecolor, Fillcolor)

ToggleColor	Boolsche Variable, die die Farbe zwischen 'Color' (FALSE) und 'Alarmcolor' (TRUE) hin- und herschaltet.	R,rR,E,P,L,pL,C,F,Bu,Im Ta
Color	DWORD-Variable, die die Füllfarbe des Elements definiert (Achtung: dies überschreibt den Wert, der gerade durch "Colors/Normal state" gesetzt ist). Die Farbe "Color" wird verwendet, wenn die Variable, die in 'ToggleColor' angegeben ist (s.o.) den Wert FALSE hat.	L,Bu,Im
Alarmcolor	DWORD-Variable, die die Füllfarbe des Elements definiert (Achtung: dies überschreibt den Wert, der gerade durch "Colors/Alarm state" gesetzt ist). Die Farbe "Alarmcolor" wird verwendet, wenn die Variable, die in 'ToggleColor' angegeben ist (s.o.) den Wert TRUE hat.	L,Bu,Im
Normalstate		
Framecolor	Rahmenfarbe des Elements im Normalzustand, zu definieren wie oben für 'Color' beschrieben	R,rR,E,P,L,pL,C Ta
Fillcolor	Füllfarbe des Elements im Normalzustand; zu definieren wie oben für 'Color' beschrieben	R,rR,E,P,L,pL,C Ta
Alarmstate		
Framecolor	Rahmenfarbe des Elements im Alarmzustand; zu definieren wie oben für 'Alarmcolor' beschrieben	R,rR,E,P,L,pL,C Ta
Fillcolor	Füllfarbe des Elements im Alarmzustand; zu definieren wie oben für 'Alarmcolor' beschrieben	R,rR,E,P,L,pL,C Ta

Look variables Aussehen-Variablen, Linienstärke, Füllattribute, Rahmenattribute (Linewidth, Fillattributes, Frameattributes)

Dies sind dynamische Definitionen für das Aussehen der Rahmenlinien und der Füllung des Elements.

(Zur statischen Definition siehe Kategorie 'Elementlook'.)

Linewidth	Integer-Variable, die die Linienstärke des Elements in Pixel definiert.	R,rR,E,P,pL,C,F,Im Ta
Fillattributes	DWORD-Variable, die die Füllung des Elements definiert. Die aktuell durch Farbvariablen definierte Farbe kann verwendet oder nicht verwendet werden: 0 = Farbe verwenden >0 = Farbe nicht verwenden	R,rR,E,P,pL,C,F,Im Ta
Frameattributes	DWORD-Variable, die die Rahmenlinien des Elements definiertt: Mögliche Typen: 0 = voll, PS_SOLID ____ 1 = gestrichelt, PS_DASH _ _ _ 2 = gepunktet, PS_DOT . . . 3 = Strich-Punkt, PS_DASHDOT _ . _ . 4 = Strich-Punkt-Punkt, PS_DASHDOTDOT _ . . _ . . 8 = unsichtbar, PS_HOLLOW	R,rR,E,P,pL,C,F,Im Ta

State variables Status-Variablen (Invisible, Deactivate inputs)

Dies sind dynamische Definitionen zur Verfügbarkeit des Elements im Online-Modus.

Invisible	Boolsche Variable; Wenn diese TRUE liefert, ist das Element im Online-Modus unsichtbar.	R,rR,E,P,L,pL,C,F,BuR Ta,Sc
Deactivate inputs	Boolsche Variable; Wenn diese TRUE liefert, bleiben Eingaben auf das Element ohne Effekt.	R,rR,E,P,L,pL,C,F,Bu Ta,Sc

m_pLineWidthVariable Linienstärken-Variable (nur für Line: IntValue)

Dynamische Definition der Liniendicke:

IntValue	Integer-Variable, die die Liniendicke in Pixel angibt. "0" entspricht "1" Dies entspricht der fixen Einstellung "LineWidth" in Kategorie "Elementlook".	L

m_pPenStyleVariable Linientyp-Variable (nur für Line: IntValue)

Dynamische Definition des Linientyps:

IntValue	Integer-Variable, die den Linientyp definiert. Mögliche Werte: 0 = voll, PS_SOLID _____ 1 = gestrichelt, PS_DASH _ _ _ 2 = gepunktet, PS_DOT . . . 3 = strich-punkt, PS_DASHDOT _ . _ . 4 = strich-punkt-punkt, PS_DASHDOTDOT _ . . _ . . 8 = unsichtbar, PS_HOLLOW This corresponds to the fix setting LineStyle in category "Elementlook".	L

m_pLineDirectionVariable Linienneigungs-Variable (nur für Line: DigitalVar)

Dynamische Definition der Neigung der Linie:

DigitalVar	Boolsche Variable, die angibt, ob die Linie von oben links nach unten rechts abfallen soll (TRUE) oder von unten links nach rechts oben ansteigen soll (FALSE, Default). Dies entspricht der fixen Einstellung "FromTopLeft" in Kategorie "Elementlook".	L

Selection (nur für Table: Selection color, Apply to columns, Selection type, Frame around selectes cells, Variable for selection X, Variable for selection Y, Variable for valid selection X, Variable for valid selection Y)

Selection		
Selection color	Füllfarbe für ausgewählte Tabellenelemente	
Apply to columns	durch Drücken dieser Taste wird die in Templates festgelegte "Alarmstate-Fillcolor", die normalerweise bei Auswahl von Elementen angezeigt würde, überschrieben	
Selection type	definiert welcher Teil der Tabelle durch ein Klicken auf eine ihrer Zellen ausgewählt wird: SEL_NONE (keine Auswahl) SEL_CELL (nur die Zelle wird ausgewählt) SEL_ROW (Auswahl der Zeile, welche die Zelle enthält) SEL_COLUMN (Auswahl der Spalte, welche die Zelle enthält) SEL_ROW_AND_COLUMN (Auswahl von Zeile und Spalte, welche die Zelle enthalten)	Ta
Frame around selected cells	Kontrollkästchen zum Ein- oder Ausschalten der Ainzeige eines Rahmens um die selektierten Elemente	Ta
Variable for selection X	Variable vom Typ INT, die auf den Array Index gesetzt wird, der zur Reihe des selektierten Elements gehört	Ta
Variable for selection Y	Variable vom Typ INT, die auf den Array Index gesetzt wird, der zur Spalte des selektierten Elements gehört; im Fall einer Struktur / POU werden die Komponenten beginnend bei 0 der Reihenfolge nach durchgezählt (Hinweis: beachten Sie, dass der Wert nur dann die korrekte Position im Array repräsentiert, wenn keine Spalten von der Anzeige in der Tabelle ausgenommen wurden)	Ta
Variable for valid selection X	Boolesche Variable, die entsprechend der Gültigkeit der selektierten Spalte auf TRUE oder FALSE gesetzt wird	Ta
Variable for valid selection Y	Boolesche Variable, die entsprechend der Gültigkeit der selektierten Zeile auf TRUE oder FALSE gesetzt wird	Ta

Buttonstate variable Schaltflächenstatus-Variable (nur für Button)

DigitalVar	Boolsche Variable, die bestimmt, ob das Schaltflächen-Element als "gedrückt" (TRUE) bzw. "nicht gedrückt" (FALSE) dargestellt wird.	Bu

Bitmap ID variable Bilddatei, Bild-ID (nur für Button)

Dynamische Definition, welches Bild im Schaltflächen-Element angezeigt werden soll.

(Für eine statische Definition siehe Kategorie 'Bitmapinfo'.)

Bitmap ID	String-Variable, die die ID der gewünschten Bilddatei, wie in der entsprechenden Bildersammlung definiert, liefert. Siehe auch die Beschreibung zu Kategorie 'Bitmapinfo'.	Bu

Isotropic Isotropisch (nur für Frame)

Isotropic	siehe oben bei 'Static ID'	F

Clipping Abschneiden (nur für Frame)

Clipping	siehe oben bei 'Static ID'	F

References Frame-Visualisierungen (nur für Frame: References, visualizations)

References		**F**
visualizations	Die Visualisierungen, die dem Frame über den Dialog 'Konfiguration der Framevisualisierungen' zugewiesen wurden, werden hier aufgelistet; **keine Eingabe möglich!**	

Eingaben Benutzereingaben (OnMouse-Aktionen: OnMouseClick, OnMouseDown, OnMouseLeave, OnMouseEnter, OnMouseUp, OnMouseMove, Toggle: Variable, Toggle on up if captured, Tap: Variable, Tap False, Tap on enter if captured) - Mausaktionen:: Variable schreiben (Eingabefeld, Numpad, Keypad), Internes Kommando (Programme ausführen, Drucken, Rezepturverwaltung), Sprachumschaltung, Visualisierungswechsel, ST-Code ausführen, Framevisualisierung umschalten, Variable umschalten, Dialog öffnen, Dialog schließen, Hotkey)

Hier wird definiert, was geschehen soll, wenn der Anwender im Online-Modus mit der Maus Eingaben auf das jeweilige Element macht.

Konfiguriert werden können die"OnDialogClosed"-Aktion, "OnMouse"-Aktionen, das "Togglen" bzw. "Tappen" von Variable, sowie ein Tastaturkürzel.

"OnDialogClosed":

Diese Eingabe erfolgt durch das Schließen eines Dialogs innerhalb der Visualisierung, der zuvor für eine Benutzereingabe geöffnet worden war. Sie können eine oder mehrere der unten beschriebenen Folge-Aktionen festlegen. Sehen Sie unten ebenfalls die Beschreibungen der Folgeaktionen "Dialog öffnen" und "Dialog schließen", die verwendet werden können, um einen Eingabedialog über eine Mausaktion auf ein Visualisierungselement zu öffnen und zu schließen.

"OnMouse"-Aktionen:

Hier kann jeder der verschiedenen Maus-Aktionen eine oder mehrere der unten definierten Folgeaktionen zugeordnet werden: Solange noch keine Folgeaktion definiert ist, erscheint "Konfigurieren" im Eigenschaftenfeld. Durch einen Mausklick auf dieses Feld wird der Dialog 'Eingabekonfiguration' geöffnet, in dem die Folgeaktion(en) konfiguriert werden können (s.u.). Bereits zugeordnete(n) Folgeaktion(en) werden im Eigenschaften-Dialog unterhalb der Elementeigenschaft angezeigt.

Toggle, Tap:

Hier werden boolsche Projektvariablen angegeben, die durch Mausklick bzw. Klick und Loslassen auf das Element den Wert TRUE oder FALSE bekommen sollen.

Hotkey:

Hier kann eine Taste mit einer bestimmten Aktion (MouseDown, MouseUp) verknüpft werden, die bei einer entsprechenden Bewegung der Taste (KeyDown, KeyUp) ausgeführt werden soll. Standardmäßig wird bei KeyDown (Taste drücken) die MouseDown Aktion ausgeführt und bei KeyUp (Taste loslassen) die MouseUp Aktion. Das kann nützlich sein, wenn eine Visualisierung sowohl über Maus- als auch Tastatureingaben bedient werden soll, weil die Eingabeaktionen dann nur einmalig konfiguriert werden müssen. Diese Tastenkonfiguration für ein Element wird auch im Tastaturkonfigurations-Editor der Visualisierung verwaltet. Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Sehen Sie die folgenden Konfigurationsmöglichkeiten:

OnDialogClosed	definiert, was geschehen soll, wenn ein Dialog, der zuvor über die Eingabe auf ein Visuailsierungselement geöffnet worden war, wieder geschlossen wird. Beispielsweise könnte mit der Option "ST Code ausführen" eine bestimmte Reaktion auf das "Resultat" des zuletzt geschlossenen Dialogs implementiert werden (dieses Resultat, z.B. "OK" oder "Cancel", wird in der"Dialog schließen" Eingabekonfiguration eines Elements festgelegt) ; Hinweis: Die Eigenschaft "OnDialogClosed" ist nicht auf das Element, für das es konfiguriert ist, beschränkt, sondern innerhalb der gesamten Visualisieurng gültig, d.h. reagiert auf jede Dialog-Schließen-Aktion! Momentan gibt es noch keine Möglichkeit, eine solche Eigenschaft für die gesamte Visualisierung zu definieren und daher muss sie einem ihrer Elemente zugewiesen werden. Klicken Sie "Eingabedialoge" für weitere Informationen und ein Beispiel bezüglich der Verwendung und Auswertung von benutzerdefinierten Dialogen.	R,rR,E,P,L,pL,C,F,Bu,Im Te
OnMouse-Aktionen		
OnMouseClick	Mausaktion: Wenn der Cursor auf das Element zeigt, wird die Maustaste gedrückt und sofort wieder losgelassen. *	R,rR,E,P,L,pL,C,F,Bu,Im Te
OnMouseDown	Mausaktion: Maustaste wird gedrückt, wenn der Cursor auf das Element zeigt *	R,rR,E,P,L,pL,C,F,Bu,Im Te
OnMouseLeave	Mausaktion: Die gedrückte Maustaste wird losgelassen solange der Cursor noch auf das Element zeigt *	R,rR,E,P,L,pL,C,F,Bu,Im Te
OnMouseEnter	Mausaktion: Der Mauszeiger wird auf das Element gezogen	R,rR,E,P,L,pL,C,F,Bu,Im Te
OnMouseUp	Mausaktion: Die Maustaste wurde gedrückt, solange der Cursor nicht auf das Element gezeigt hatte und wird losgelassen nachdem der Cursor auf das Element gezogen wurde *	R,rR,E,P,L,pL,C,F,Bu,Im Te
OnMouseMove	Mausaktion: Die Maus wird bewegt *	R,rR,E,P,L,pL,C,F,Bu,Im Te

Toggle		
Variable	Hier kann eine boolsche Variable eingetragen werden, die bei jedem Mausklick auf das Element zwischen TRUE und FALSE wechseln soll. Die Eingabehilfe kann über Schaltfläche ... geöffnet werden, die nach einem Doppelklick auf das Feld bereitsteht.	R,rR,E,P,L,pL,C,F,Bu,Im Te
Toggle on up if captured	Wenn diese Option deaktiviert ist (Default): Wenn die Maustaste gedrückt wird während der Cursor auf das Element zeigt, dann aber der Cursor aus dem Element gezogen wird, bevor die Taste wieder losgelassen wird, wird die zugeordnete Variable s.o.) nicht getoggelt. So kann also eine begonnene Toggle-Eingabe abgebrochen werden. Wenn die Option aktiviert ist: Die zugewiesene boolsche Variable wird auch getoggelt, wenn die gedrückte Maustaste losgelassen wird nachdem der Cursor aus dem Element herausbewegt wurde.	R,rR,E,P,L,pL,C,F,Bu,Im
Tap		
Variable	Hier kann eine boolsche Variable eingetragen werden, die auf TRUE gesetzt wird, wenn die Maustaste gedrückt wird, wenn der Cursor auf das Element zeigt, und die wieder FALSE wird, wenn die Maustaste wieder losgelassen wird.	R,rR,E,P,L,pL,C,F,Bu,Im Te
Tap False	Wenn diese Option aktiviert ist, wird das oben für die eingetragene Variable beschriebene Tap-Verhalten umgekehrt. D.h. wenn die Maustaste gedrückt wird, wird die Variable auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird die Variable auf TRUE gesetzt.	R,rR,E,P,L,pL,C,F,Bu,Im Te
Tap on enter if captured	Wenn diese Option aktiviert ist, wird die oben eingetragene Variable auch "getapped", wenn die Maustaste gedrückt wurde, bevor der Cursor auf das Element gezogen wurde.	R,rR,E,P,L,pL,C,F,Bu,Im Te

Hotkey		R,rR,E,P,L,pL,C,F,Bu Te
Taste	Tastenname, z.B. "M"; eine Auswahlliste stellt die aktuell unterstützten Tasten bereit	
Eingaben	Aktion, die ausgeführt werden soll, wenn die Taste + ev. definierte Modifizierer (Umschalten, Steuerung, Alt) verwendet wird; mögliche Werte, die in einer Auswahlliste bereitstehen: - Keine Aktion - MouseDown Aktion (OnMouseDown Aktion, wenn die Taste gedrückt wird) - MouseUp Aktion (OnMouseUp Aktion, wenn die Taste losgelassen wird) - MouseDown/MouseUp Aktion (OnMouseUp und OnMouseDown Aktione, wenn die Taste gedrückt oder losgelassen wird)	
Umschalten	wenn diese Option aktiviert ist, muss die Taste mit der Umschalt-Taste kombiniert verwendet werden	
Steuerung	wenn diese Option aktiviert ist, muss die Taste mit der Strg-Taste kombiniert verwendet werden	
Alt	wenn diese Option aktiviert ist, muss die Taste mit der Alt-Taste kombiniert verwendet werden	

*** "OnMouse..."-Folgeaktionen:**

Die nachfolgend beschriebenen Folgeaktionen können im Dialog 'Eingabekonfiguration' für eine bestimmte Mausaktion auf das Element konfiguriert werden. Der Dialog öffnet nach einem Mausklick auf das Eigenschaftenfeld im Element-Eigenschaften-Dialog.

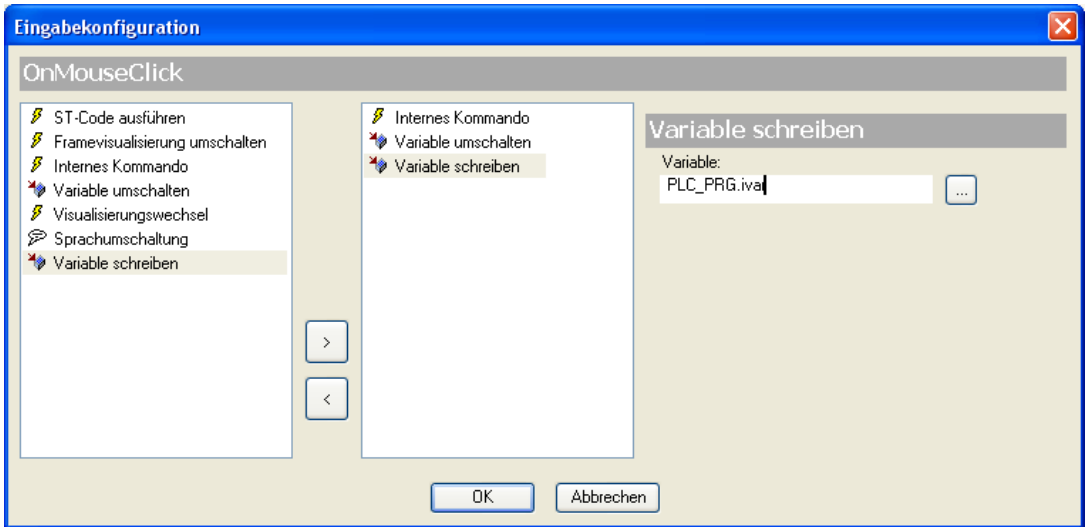


Abb. 432 Dialog 'Eingabekonfiguration'

Ganz links stehen die möglichen Folge-Aktionen zur Auwahl. In der Mitte stehen die für die aktuelle Mausaktion bereits definierten Folge-Aktionen. Ganz rechts erscheinen immer die für die im mittleren Feld ausgewählte Aktion relevanten Konfigurationsfelder.

Um eine oder mehrere der Folge-Aktionen für die aktuelle Mausaktion zu definieren, wählen Sie sie im linken Feld aus und betätigen die Schaltfläche mit dem Rechtspfeil, um sie in das rechte Feld zu bringen. Wählen Sie dann eine Aktion im mittleren Feld aus, um die entsprechende Konfiguration in den Feldern auf der rechten Seite durchzuführen.

Sehen Sie im folgenden die Konfigurationsmöglichkeiten für die einzelnen Folge-Aktionen:

-Dialog schließen

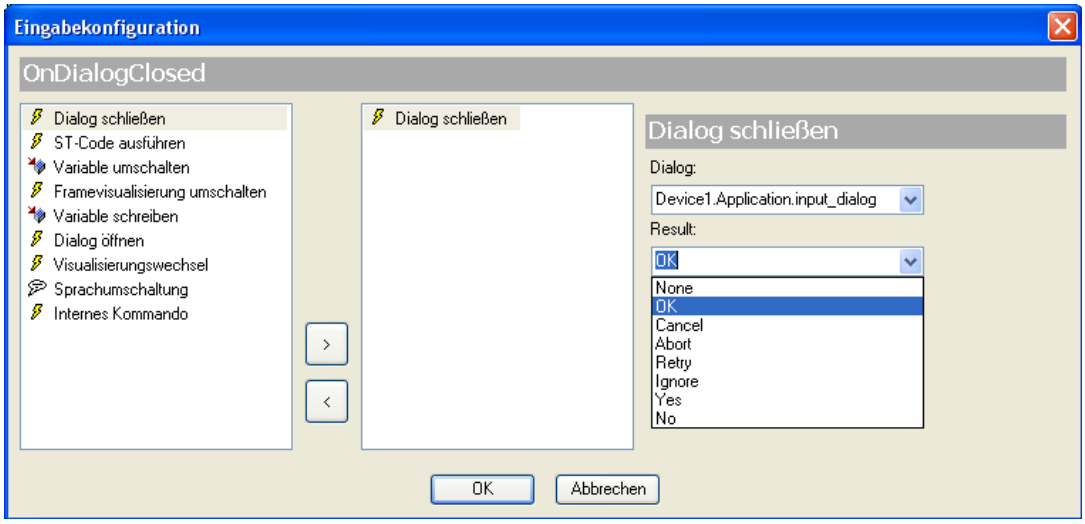


Abb. 433 Eingabekonfiguration, Dialog schließen

Hier kann festgelegt werden, dass auf die Mausaktion hin der angegebene Dialog mit dem angegebenen Resultat geschlossen wird. Wählen Sie den gewünschten Dialog von der Auswahlliste, die - wie in der 'Dialog öffnen' Konfiguration (siehe unten) - alle momentan verfügbaren Dialog-Visualisierungen anbietet.

Die 'Result' Liste bietet die Standardoptionen, die in Dialogen verwendet werden, in denen eine Benutzerreaktion gefordert wird: OK, Cancel, Abort, Retry, Ignore, Yes, No.

Beachten Sie, dass das Resultat, das vom zuletzt geschlossenen Dialog stammt, abgerufen werden kann und dass eine entsprechende Reaktion darauf in einem beliebigen Element derselben Visualisierung konfiguriert werden kann. Verwenden Sie dazu die Konfigurationsoption 'OnDialogClosed'.

- ST-Code ausführen

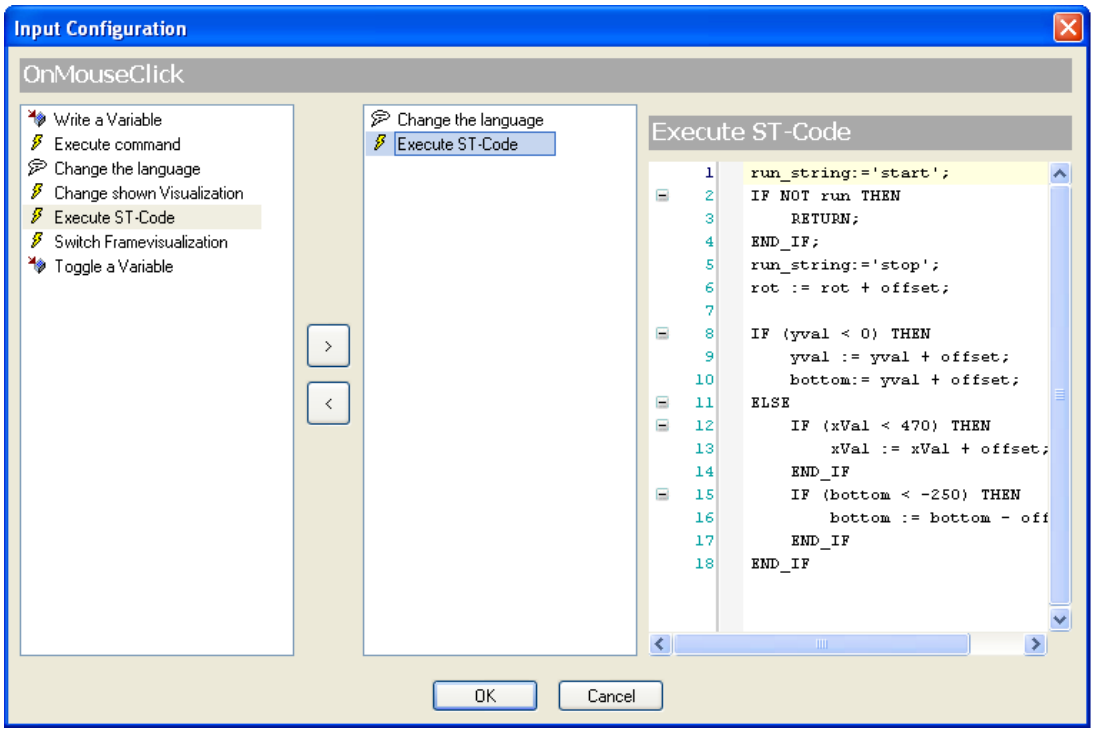


Abb. 434 Eingabekonfiguration, ST-Code ausführen

Im Eingabefeld kann Code in Strukturiertem Text eingegeben werden, der auf die Mausaktion hin ausgeführt werden soll.

- Variable umschalten

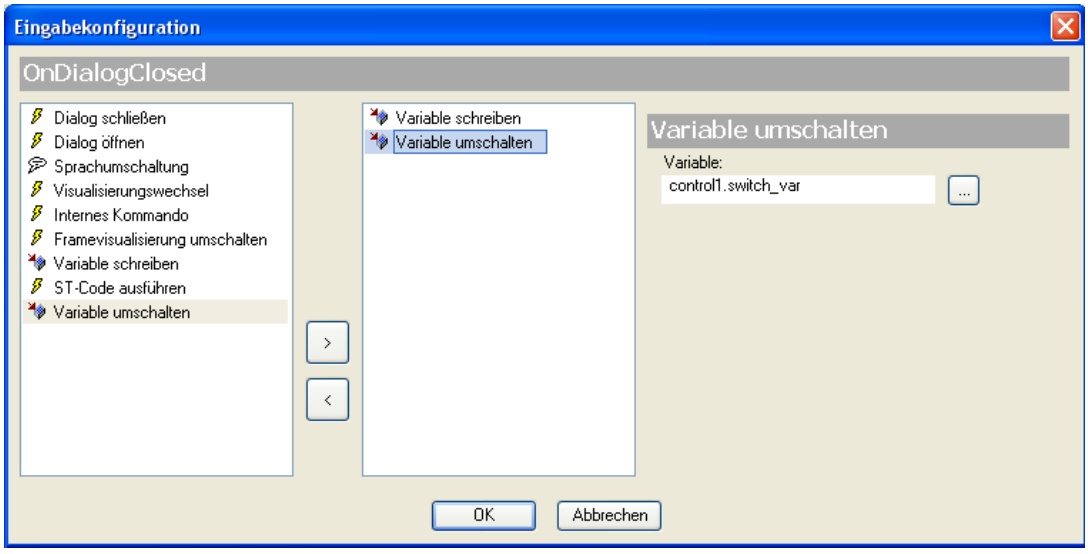


Abb. 435 Eingabekonfiguration, Variable umschalten

Hier kann eine boolsche Variable eingegeben werden, die bei wiederholter Mausaktion zwischen TRUE und FALSE wechseln soll. Beispiel: "plc_prg.bvar1"

- Frame-Visualisierung umschalten

Voraussetzungen: In einer Visualisierung im Projekt gibt es Frame-Elemente, denen über den Befehl 'Frame-Auswahl' und Dialog 'Konfiguration der Framevisualisierungen' verschiedene Visualisierungen zugeordnet wurden. Diese Visualisierungen erhalten innerhalb des Frames eine Indizierung 0, 1, 2, usw. und im Standardfall wird die erste der zugeordneten Visualisierungen (Index "0") online im Frame angezeigt.

Frame-Visualisierung umschalten:

Im vorliegenden Dialog können Sie nun konfigurieren, dass bei Ausführen der Mausaktion auf das aktuelle Visu-Element in einem bestimmten Frame-Element eine bestimmte der ihm zugeordneten Visualisierungen angezeigt wird. Mit dem aktuellen Element kann also ein "Umschalter" für die Anzeige von Visualisierungen in einem Frame programmiert werden.

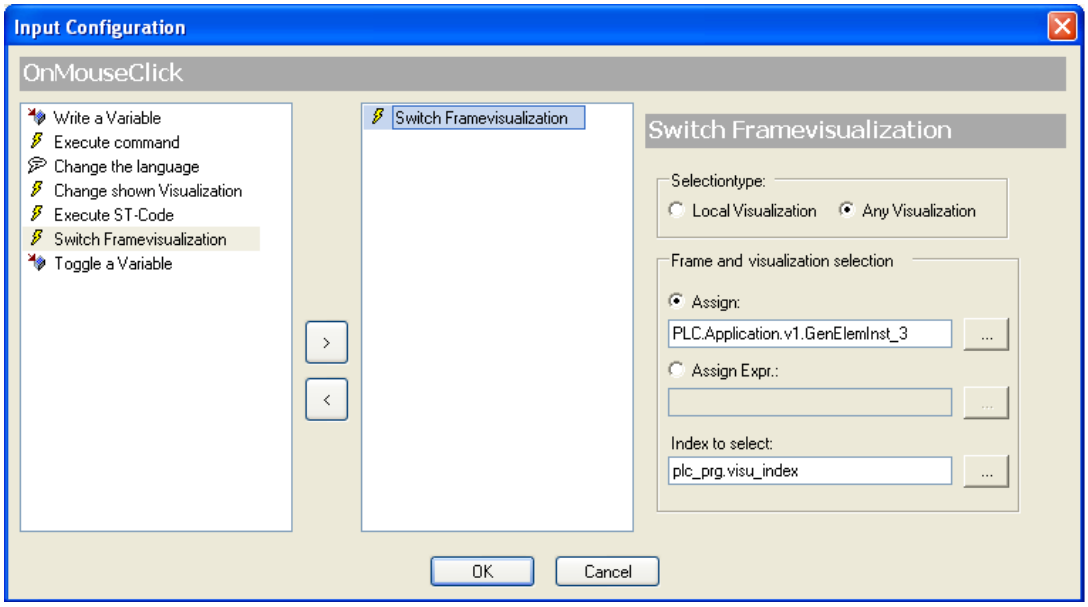


Abb. 436 Eingabekonfiguration, Frame-Visualisierung umschalten

Auswahlart:

Die Auswahl des betroffenen Frame-Elementes kann sich auf die aktuelle Visualisierung beschränken, was einen einfacheren Konfigurationsdialog, allerdings ohne dynamische Konfigurationsmöglichkeit, bedeutet -> Option 'Lokale Visualisierung',

oder alle Visualisierungen des Projekts einbeziehen, was auch dynamische Frame- und Visu-Angaben erlaubt -> Optin 'Beliebige Visualisierung'.

Lokale Visualisierung: Nur Frames der aktuellen Visualisierung können angesprochen werden und stehen somit hier im Feld.'Auswahl des Frames' zur Auswahl (s.u.). Dies ist ein einfacher Dialog zur schnellen direkten Konfiguration in der lokalen Visualisierung, bietet allerdings nicht die Möglichkeit, die gewünschte Visualisierung über eine Projektvariablen anzugeben. Wenn dies nötig ist, wählen Sie Option 'Beliebige Visualisierung', siehe unten.

Das Feld 'Auswahl des Frames' zeigt die lokalen Frame-Elemente und eingerückt darunter jeweils die zugeordneten Visualisierungen.

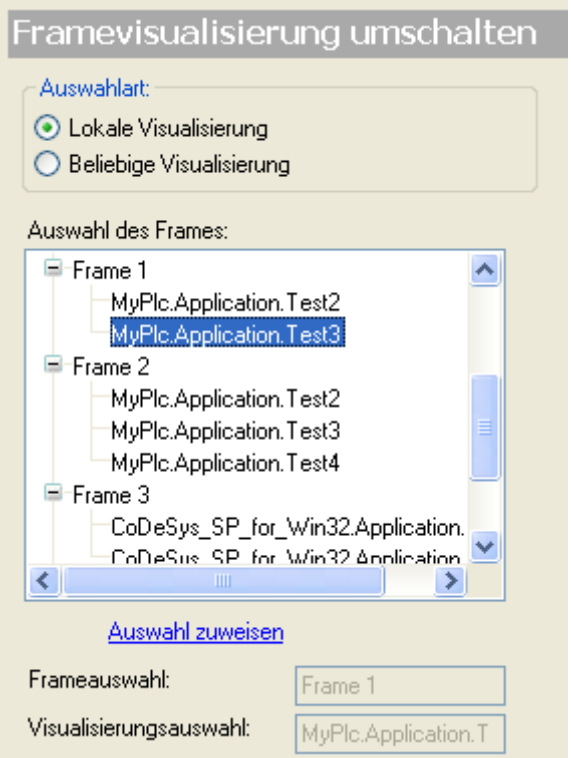


Abb. 437 Auswahl des Frames, lokal

Wählen Sie unterhalb des betroffenen Frames die gewünschte Visualisierung, die auf die Mausaktion hin angezeigt werden soll. Klicken Sie auf 'Auswahl zuweisen', um die Einstellung zu speichern. Die aktuelle Auswahl wird daraufhin in den Feldern Frameauswahl und Visualisierungsauswahl angezeigt.

Beliebige Visualisierung: Alle Frames des Projekts und ihre zugeordneten Visualisierungen stehen zur Auswahl. In diesem Fall können Frame und Visualisierung auch über Projektvariablen, also dynamisch angegeben werden. Die 'Auswahl von Frame und Visualisierung' wird dann über folgende Felder und Optionen vorgenommen:

Das gewünschte Frame-Element kann direkt oder über einen Ausdruck angegeben werden:

Direkte Zuweisung: Wenn diese Option aktiviert ist, muss der komplette Pfad des betroffenen Frame-Elementes direkt eingegeben werden. Über Schaltfläche [...] kann die Eingabehilfe dazu verwendet werden. Syntax: <Gerätename>.<Applikationsname>.<Visualisierungsname>.<Frame-Elementname>. Sehen Sie ein Beispiel in der folgenden Abbildung.

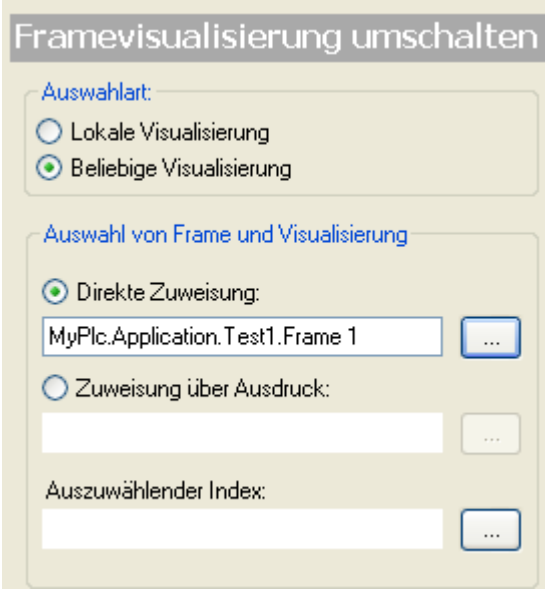


Abb. 438 Auswahl des Frames, direkt

Zuweisung über Ausdruck: Diese Option kann alternativ aktiviert werden, um eine Projektvariable vom Typ STRING für die Angabe des Frame-Elements zu verwenden. Über Schaltfläche [...] kann die Eingabehilfe geöffnet werden. Die Variable muss dann den kompletten Pfad des Frame-Elementes liefern. Syntax: <Gerätename>.<Applikationsname>.<Visualisierungsname>.<Frame-Elementname>. Beispiel: frame_var: STRING:='MyPlc.Application.Test1.Frame1';

Die gewünschte Visualisierung, die bei Mausaktion im ausgewählten Frame angezeigt werden soll, wird über ihren Index angegeben. Dieser Index ist eine mit 0 beginnende aufsteigende, ganzzahlige Nummerierung der Visualisierungen, die einem Frame im Dialog 'Konfiguration der Framevisualisierungen' zugewiesen wurden. Somit wird initial standardmäßig immer die erste Visualisierung in dieser Liste angezeigt.

Auszuwählender Index: Geben Sie hier den Index der gewünschten Visualisierung direkt oder über eine in der Applikation verwendete Projektvariable an. Die Schaltfläche [...] öffnet die Eingabehilfe zur Auswahl einer Variablen.

- Variable schreiben (Eingabefeld, Numpad, Keypad)

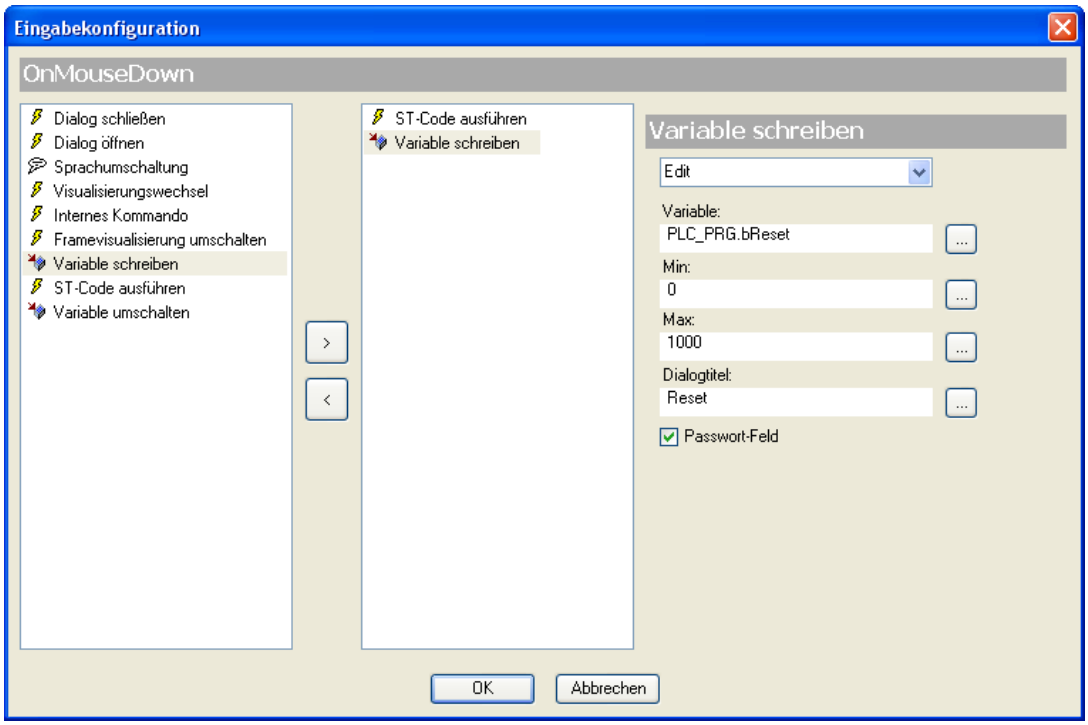


Abb. 439 Eingabekonfiguration, Variable schreiben

Wenn für ein Visualisierungselement eine Eingabekonfiguration des Typs 'Variable schreiben' gibt, dann wird - sobald die entsprechende Mausaktion ausgeführt wird - das Element eine Möglichkeit bieten, einen Wert einzugeben. Der Wert kann dann als eine Zeichenfolge oder über ein Numpad oder Keypad eingegeben werden und wird nach Beendigung der Eingabe auf die Projektvariable geschrieben, die hier im Eingabekonfigurationsdialog angegeben ist. Der Wert wird entsprechend des Datentyps der Projektvariable als Text oder als numerischer Wert interpretiert.

Verwenden des Konfigurationsdialogs:

Wählen Sie von der Auswahlliste im rechten Teil des Dialogs einen der folgenden Eingabetypen:

Edit: Die Mausaktion wird ein Eingabefeld öffnen, in das ein Text oder ein numerischer Wert eingetippt werden kann.

VisuDialogs.Keypad: Die Mausaktion wird ein simuliertes Keypad (Tastaturfeld) öffnen. Sie können durch Mausklicks auf die entsprechenden Tastenfelder eine Zeichenfolge eingeben.

VisuDialogs.Numpad: Die Mausaktion wird ein simuliertes Numpad (Nummerntastaturfeld) öffnen. Sie können durch Mausklicks auf die entsprechenden Tastenfelder eine numerische Zeichenfolge eingeben.

Zusätzlich: Wenn es benutzerdefinierte Visualisierungen im Projekt gibt, die in ihren Eigenschaften als "numpad/keypad" definiert sind:
<device>.<application>.<numpad/keypad-visualization>: Die Mausaktion wird das Numpad bzw. Keypad öffnen, das in der gewählten Visualisierung dargestellt wird. Beachten Sie, dass für diesen Zweck die Option Bilder ohne Pfad registrieren aktiviert sein muss, damit die benötigten Hintergund-Bitmaps geladen werden können.

Hinweis: Zur Benutzung von Keypad oder Numpad muss die Bibliothek VisuDialogs.library in den Bibliotheksmanager integriert werden.

In der Box mit dem Titel **Zu bearbeitende Variable auswählen** kann festgelegt werden ob der vom Benutzer eingegebene Wert auf die Text Ausgangsvariable geschrieben werden soll oder auf eine andere Variable im Projekt. Die Text Variable ist Teil der Elementeigenschaft Text variables und wird einfach durch Anklicken selektiert. Möchten Sie den Wert auf eine andere Projekt Variable schreiben, so müssen Sie deren Namen (einschließlich des Pfads) eingeben. Zu diesem Zweck kann auch der Eingabeassistent über die Taste verwendet werden. Im Feld **Variable** wird die Projektvariable angegeben, auf die der vom Benutzer eingegebene Wert geschrieben werden soll. Die Eingabehilfe kann dazu über die Schaltfläche ... geöffnet werden.

Ein Grenzbereich für erlaubte Eingaben kann über die Eingabe des **Min**imal- und des **Max**imalwerts, als Wert oder über Projektvariablen, definiert werden.

Dialog Titel: Hier kann als Text oder über eine Textvariable angegeben werden, was in der Titelleiste des Dialogs angezeigt werden soll.

Falls Sie wünschen, dass Ihre Eingabe versteckt dargestellt werden soll durch "**********" (wie im Fall von Passworten) so aktivieren Sie das Kontrollkästchen vor **Passwort-Feld**.

Beispiel:

Sehen Sie als Beispiel die obige Dialogabbildung und nehmen Sie an, es handele sich um den Konfigurationsdialog für ein Rechteck-Element. Wenn im Online-Modus die Maustaste auf diesem Rechteck gedrückt wird, erscheint ein Nummerntastaturfeld mit dem Titel, der von der Variablen PLC_PRG. MyTitle geliefert wird.

Durch Mausklicks auf 1, 2 und 3 kann beispielsweise der Wert "123" eingegeben werden. Dieser Wert wird im oberen Teil des Dialogs angezeigt, ebenso wie die Minimal- und Maximalwerte für die Eingabe, die durch PLC_PRG.minVal und PLC_PRG.maxVal gegeben werden. Sobald die Eingabe mit einem Mausklick auf die Taste OK bestätigt wird, wird "123" auf die Variable PLC_PRG.ivar geschrieben. Wenn "ivar" als STRING-Variable definiert ist, erhält es den Wert '123', wenn ivar eine numerische Variable ist, erhält sie den Wert 123.

- Dialog öffnen

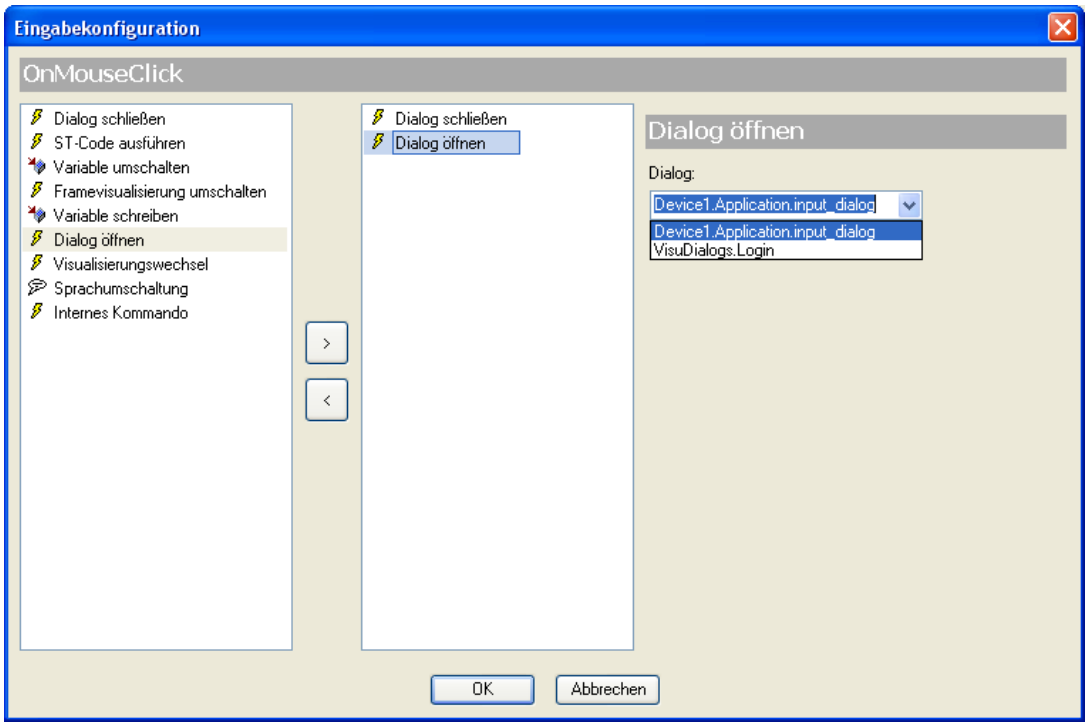


Abb. 440 Eingabekonfiguration, Dialog öffnen

Hier können Sie definieren, dass auf die Mausaktion hin ein Dialog geöffnet werden soll, der durch eine an andere (Standard- oder vom Benutzer erstellte) Visualisierung repräsentiert wird. Die Auswahlliste bietet alle Visualisierungen an, die in ihren Objekteigenschaften den Verwendungszweck "Dialog" eingetragen haben. Somit kann ein selbst-erstellter Dialog als Benutzereingabemaske in einer Visualisierung verwendet werden.

Zumindest das Standardobjekt "VisuDialogs.Login" ist verfügbar.

- Visualisierungswechsel

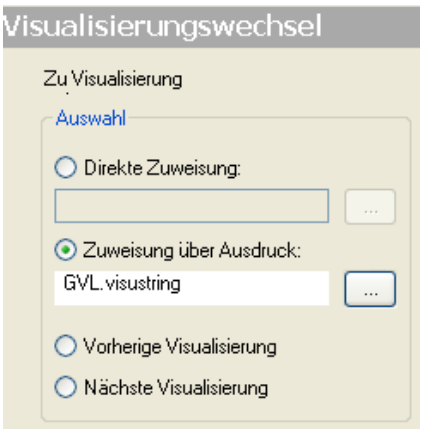


Abb. 441 Eingabekonfiguration, Visualisierungswechsel

Wählen Sie eine der folgenden Optionen zur Festlegung, welche Visualisierung auf die Mausaktion im Online-Modus hin angezeigt werden soll (Auswahl):

Direkte Zuweisung: Die Visualisierung kann direkt eingegeben werden. Der gesamte Pfad <Gerätename>. <Applikationsname>.<Visualisierungsname> muss eingetragen werden. Dazu wird am besten die Eingabehilfe verwendet, die über Schaltfläche [...] geöffnet werden kann. Beispiel: "MyPlc.Appl1.Visu_xy"

Zuweisung über Ausdruck: Hier kann eine Variable vom Typ STRING angegeben werden, die von der Applikation verwendet wird und die den Namen der Visualisierung liefert. Beispiel: "visu_stringvar: STRING := 'MyPlc.Appl1.Visu_xy' "

Die Abfolge, in der Visualisierungen nacheinander über Benutzereingaben angezeigt wurden, wird intern gespeichert. MIt den folgenden beiden Optionen können diese Informationen genutzt werden:

Vorherige Visualisierung: Die zuvor angezeigte Visualisierung wird wieder angezeigt. Wenn zuvor noch keine aufgerufen war, wird weiterhin die aktuelle gezeigt.

Nächste Visualisierung: Die Visualisierung, die in der aufgezeichneten Abfolge der Visualisierungswechsel hinter der aktuellen steht, wird angezeigt. Dies ist also nur möglich, wenn aus dieser vorher bereits ein Visualisierungswechsel über 'Vorherige Visualisierung' vorgenommen wurde.

- Sprachumschaltung

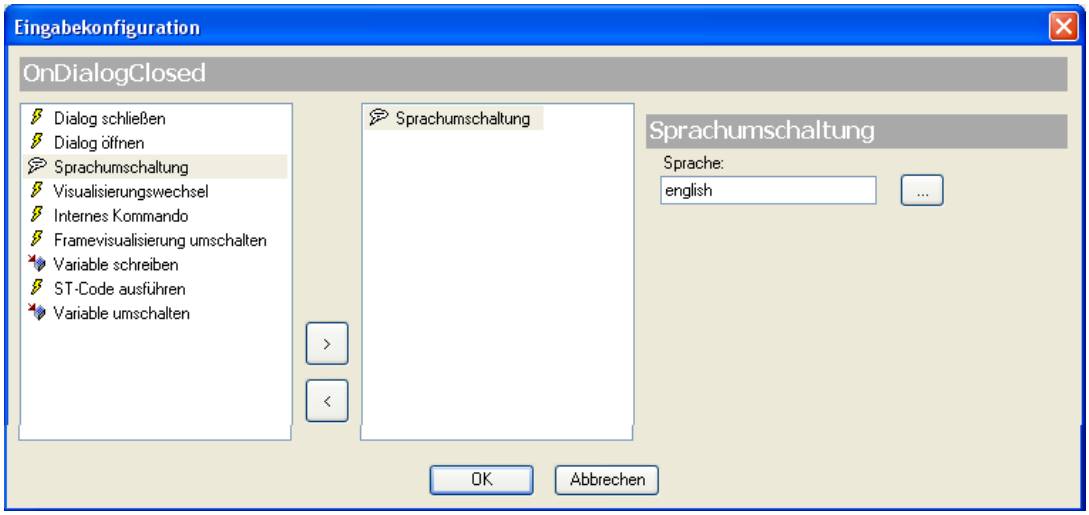


Abb. 442 Eingabekonfiguration, Sprachumschaltung

Hier kann eine Landessprache angegeben werden, die auf die Mausaktion hin für die Darstellung der Visualisierungstexte verwendet wird. Verwenden Sie dabei das Sprachkürzel, das in der/den zugehörigen Textliste(n) verwendet wird. Alternativ können Sie auf die Taste [...] klicken, um die gewünschte Sprache im Eingabeassistenten auszuwählen.

- Internes Kommando (Ausführen von Programmen, Drucken Rezepturverwaltung)

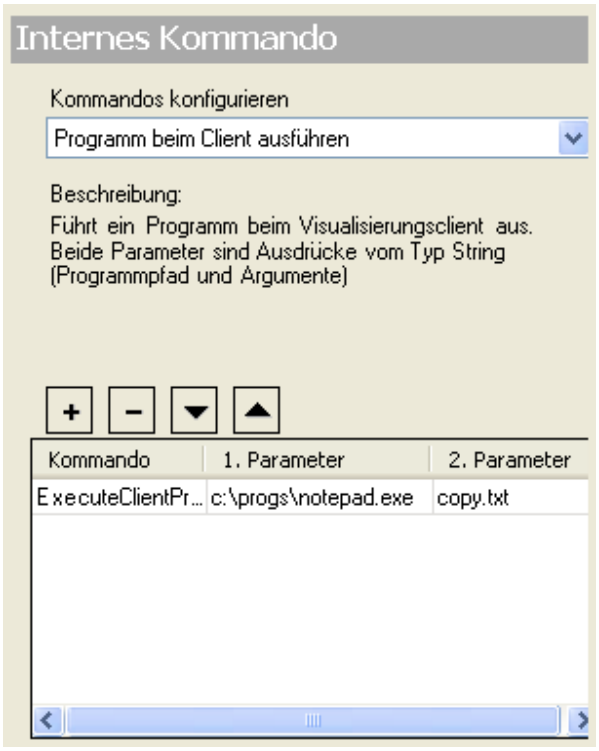
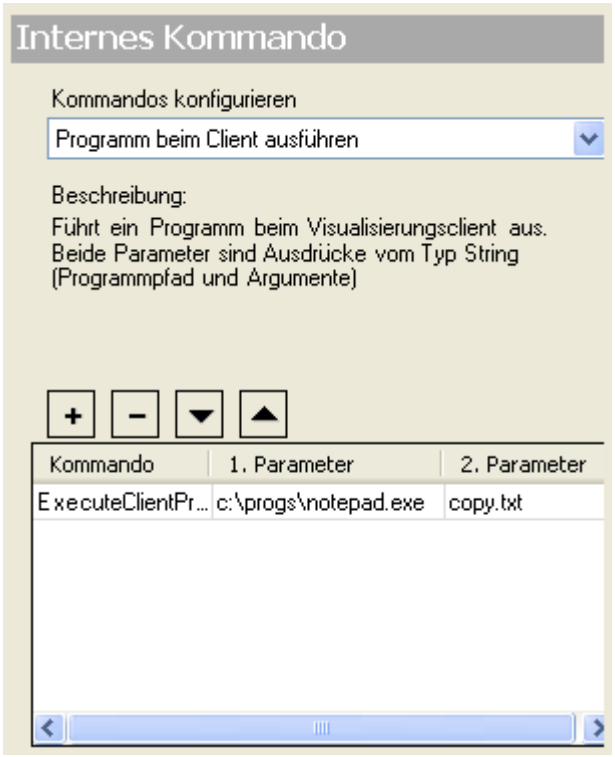


Abb. 443 *Eingabekonfiguration, Internes Kommando*



Hier können ein oder mehrere Kommandos definiert werden, die auf die Mausaktion hin ausgeführt werden sollen.

Wählen sie aus der Liste **Kommandos konfigurieren** einen Befehl aus und betätigen Sie die + Schaltfläche, um ihn in der Tabelle im unteren Teil des Dialogs einzufügen. Diese Tabelle enthält alle aktuell für die aktuelle Eingabekonfiguration ausgewählten Befehle.

Eine kurze Beschreibung zum oben in der Liste ausgewählten Befehl wird immer im Mittelteil des Dialogs angezeigt. Ergänzen Sie dementsprechend die Parameter des Befehls unten in den Tabellenspalten **1.Parameter** und **2.Parameter**. (Spalte **Kommando** zeigt den internen Kommandonamen.) Sehen Sie hierzu auch weiter unten auf dieser Hilfeseite die Beschreibung der einzelnen Kommandos.

Mit der -Schaltfläche kann ein in der Tabelle selektierter Eintrag wieder entfernt werden.

Die konfigurierten Befehle werden später, wenn die Benutzereingabe auf das Visualisierungselement erfolgt, entsprechend Ihrer Anordnung in der Tabelle von oben nach unten nacheinander ausgeführt. Um die Reihenfolge zu verändern, können die Einträge mit den Schaltflächen ▼ und ▲ verschoben werden.

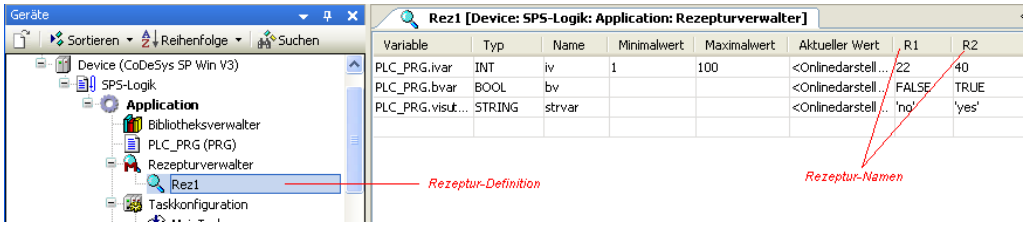
Die folgenden Kommandos stehen zur Auswahl:

Programm auf der Steuerung ausführen	Das angegebene Programm (*.exe) wird auf der Steuerung bzw. beim Visualisierungs-Client ausgeführt.
Programm beim Client ausführen	1.Parameter: Zeichenfolge (string), Pfad der Programmdatei (Beispiel: "C:\programs\notepad.exe") 2. Parameter: Zeichenfolge (string), Argumente für das auszuführende Programm, z.B. Name einer Datei, die das Programm öffnen soll (Beispiel: "copyfile.txt")
Drucken	Der Standarddialog Drucken wird geöffnet, wo Einstellungen zum Einstellen des Druckebereichs und der Druckerparameter etc. vorgenommen werden können. Darüber kann die aktuelle Visualisierung ausgedruckt werden.

Für die Kommandos zur Rezepturverwaltung beachten Sie bitte folgendes:

Die "**Rezepturdefinition**" ist ein Objekt unterhalb des Rezepturverwalters im Gerätebaum des Projekts. Sie besteht aus einer Liste von Variablen und den für diese Variablen vorhandenen Rezepturen (d.h. Variablenwerten). Sie kann im Editorfenster des Rezepturverwalters geöffnet und bearbeitet werden.

Der "**Rezepturname**" ist der Name einer in der Rezepturdefinition beschriebenen Rezeptur für die jeweilige Liste von Variablen. Die Rezepturnamen sind dort als Tabellenüberschriften eingetragen.



Rezeptur erstellen	Eine neue Rezeptur wird in der angegebenen Rezepturdefinition angelegt. 1.Parameter: Name der Rezepturdefinition 2.Parameter: Name der neuen Rezeptur
Rezeptur lesen	Die aktuellen Werte der Variablen der angegebenen Rezepturdefinition werden von der Steuerung gelesen und in die angegebene Rezeptur geschrieben. Dabei werden die Werte implizit gespeichert (in einer Datei auf der Steuerung) und gleichzeitig in der Tabelle der Rezepturdefinition im »PLC Designer« Rezepturverwalter angezeigt. In anderen Worten: Die in »PLC Designer« verwaltete Rezeptur wird mit den Werten aus der Steuerung aktualisiert. 1.Parameter: Name der Rezepturdefinition 2.Parameter: Name der Rezeptur, für die die Werte von der Steuerung gelesen und als neue Rezepturwerte gespeichert werden sollen.
Rezeptur schreiben	Die Werte der angegeben Rezeptur, wie sie im Rezepturverwalter vorliegen, werden auf die entsprechenden Variablen in der Steuerung geschrieben. 1.Parameter: Name der Rezepturdefinition 2.Parameter: Name der Rezeptur aus dieser Rezepturdefinition, deren Werte auf die Variablen in der Steuerung geschrieben werden sollen.
Rezeptur speichern	Die Werte der gegebenen Rezeptur werden in eine Datei mit Erweiterung ***.txtrecipe** gespeichert, deren Name definiert werden muss. Dazu öffnet der Standarddialog zum Speichern einer Datei. ACHTUNG: Die implizit verwendeten Rezeptur-Dateien, die als Zwischenablage beim Lesen und Schreiben von Rezepturen nötig sind, dürfen dabei nicht überschrieben werden. D.h. der Name für die neue Datei muss anders lauten als "<Rezepturname>.<Rezepturdefinitionsname>.txtrecipe" ! 1.Parameter: Name der Rezepturdefinition 2.Parameter: Name der Rezeptur, die in der angegebenen Datei gespeichert werden soll.
Rezeptur laden	Die Rezeptur, die in einer Datei gespeichert wurde (s.o. "Rezeptur speichern"), kann aus dieser Datei wieder geladen werden. Zur Auswahl der Datei wird der Standard-Dateiauswahl-Dialog geöffnet. Der Filter ist automatisch auf Dateierweiterung "*.txtrecipe" gestellt. Nach dem Laden wird die Anzeige der betreffenden Rezeptur im »PLC Designer« Rezepturverwalter aktualisiert. 1.Parameter: Name der Rezepturdefinition 2.Parameter: Name der Rezeptur, für die die Werte aus der angegebenen Datei wieder in den Rezepturverwalter geladen werden sollen.
Rezeptur löschen	Die angegebene Rezeptur wird aus der angegebenen Rezepturdefinition gelöscht. 1.Parameter: Name der Rezepturdefinition 2.Parameter: Name der Rezeptur

Jede Eingabe per Maus, der im Eingabe-Konfigurationsdialog eine Aktion zugewiesen ist, zeigt im Eigenschaften-Fenster die Eigenschaft "Konfiguriert" an.

Scrollbar spezifische Eingaben (nur für Scrollbar: Input Variable, Output Variable, Minimum value, Maximum value)

Input Variable	Variable, entsprechend deren Wert der Regler des Scrollbars positioniert wird	Sc
Output Variable	Variable, deren Wert durch Positionieren des Scrollbar-Reglers geschrieben wird	Sc
Minimum value	Wert, der einer Positionierung des Reglers ganz links entspricht	Sc
Maximum value	Wert, der einer Positionierung des Reglers ganz rechts entspricht	Sc

Definition von Farbwerten:

Eine Farbe wird durch eine hexadezimale Zahl definiert, die sich aus den Blau/Grün/Rot (RGB) Komponeten zusammensetzt.

Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar.

FF Blaue Komponente
00 Grüne Komponente
FF Rote Komponente

Beispiel: 16#FF00FF

Für **statische Definitionen** (Kategorie 'Farben' in den Elementeigenschaften) kann die Farbe aus einer Auswahlliste oder dem Standard-Farbauswahl-Dialog gewählt werden (der über Schaltfläche ... geöffnet werden kann). Beispiel: 0; 128; 64

Für **dynamische Definitionen** (Kategorie 'Farbvariablen') muss eine Projektvariable vom Typ DWORD in den Elementeigenschaften eingetragen werden, die die Farbedefinition liefert:
z.B. "prog1.dwFillColor".

Die Variable muss den Farbwert in hexadezimalem Format liefern. Die ersten beiden Nullen nach "16#" sollten in jedem Fall angegeben werden um die DWORD-Größe zu füllen.
Beispiel: dwFillColor := 16#FF00FF;

13.18.6 Schnittstellen-Editor für Visualisierung

Der Schnittstellen-Editor für Visualisierungen wird mit dem Plug-In 'Interface Editor' installiert. Er ist Teil des Visualisierungs-Editors und kann - wenn dieser aktiv ist - mit dem Befehl 'Schnittstellen Editor' (standardmäßig im Menü 'Visualisierung') aus- und eingeblendet werden. Er erscheint im oberen Teil des Visualisierungs-Editors als separates Registerblatt neben anderen Editoren wie z.B. den Editoren für die Tastaturkonfiguration oder die Elementliste.

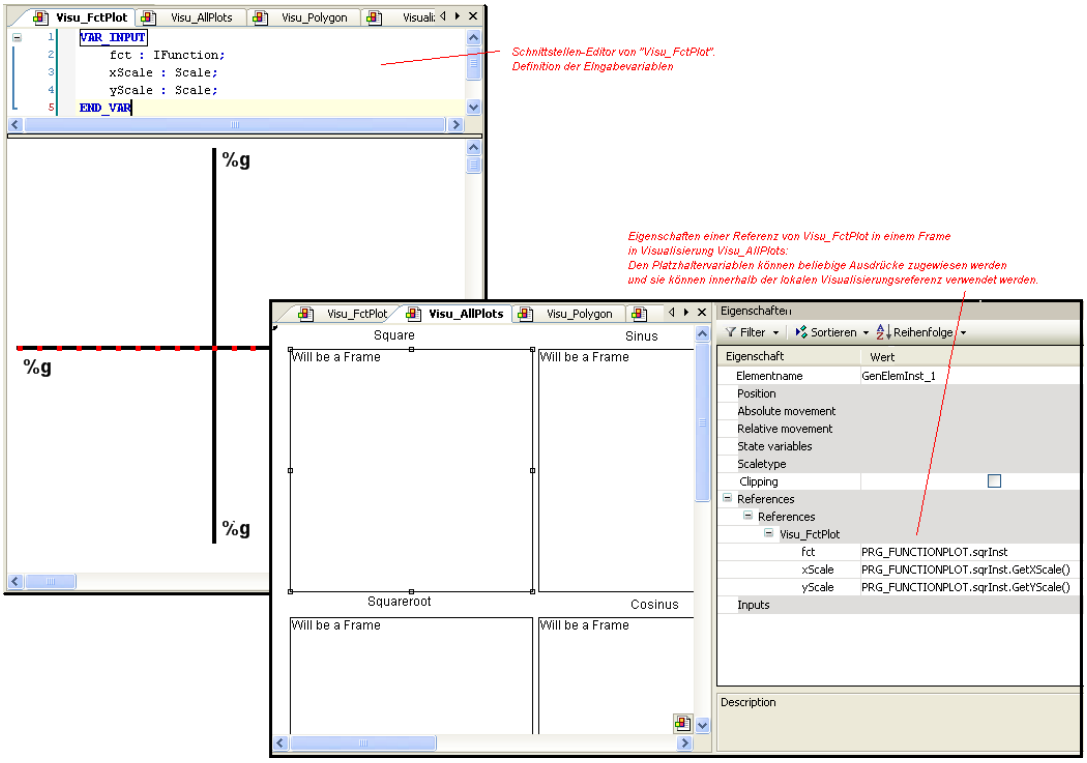


Abb. 444 Schnittstellen-Editor, Beispiel

Der Editor wird verwendet, um "**Platzhalter**"-Variablen in einer Visualisierung zu definieren, die als Referenz in eine andere Visualisierung eingefügt werden soll.

Sehen Sie Frames, Referenzen... für Informationen zu "Referenzen" und "Frames".

Sehen Sie Visualisierung für generelle Informationen (Voraussetzungen, Überblick) zur Visualisierung in »PLC Designer« V3.x.

Da eine Visualisierung als Funktionsblock behandelt wird, erscheint der Schnittstellen-Editor als normaler **Deklarationseditor** im oberen Teil des Visualisierungs-Editor-Fensters. Hier können Eingabevariablen (VAR_INPUT) deklariert werden, die später, wenn die Visualisierung in einem Frame-Element einer anderen Visualisierung eingefügt wird, durch lokal verwendete Variablen oder Ausdrücke ersetzt werden können. Diese Ersetzungen müssen im Eigenschaften-Dialog des Frame-Elements vorgenommen werden. Dort ist die eingefügte (instanzierte) Visualisierung und ihre Eingabevariablen aufgeführt und den Variablen können Ausdrücke zugewiesen werden, die bei der lokalen Konfiguration der Instanz verwendet werden können.

Wenn die Schnittstelle einer in einem Frame eingefügten Visualisierung geändert wird, wird beim Speichern oder Kompilieren des Projekts oder beim Öffnen eines betroffenen Objekts zunächst der Dialog **Aktualisierung der Frame-Parameter** geöffnet, in dem definiert wird, wie die veränderten Schnittstellenparameter behandelt werden sollen:

Sehen Sie die Verwendung dieses Dialogs an folgendem Beispiel:

Die Schnittstelle von Visualisierung *basic_visu* enthält zunächst nur den Parameter *iVisu*. *basic_visu* wird in *visu2* und *visu3* in einem Frame eingefügt, wobei *iVisu i*n *visu2* der Projektvariablen *PLC_PRG.ivar* zugewiesen ist. Das Projekt wird zunächst so verwendet. Später wird die Schnittstelle von *basic_visu* um die Paramter *bToggleColor*, *strText* und *bVisu* erweitert:

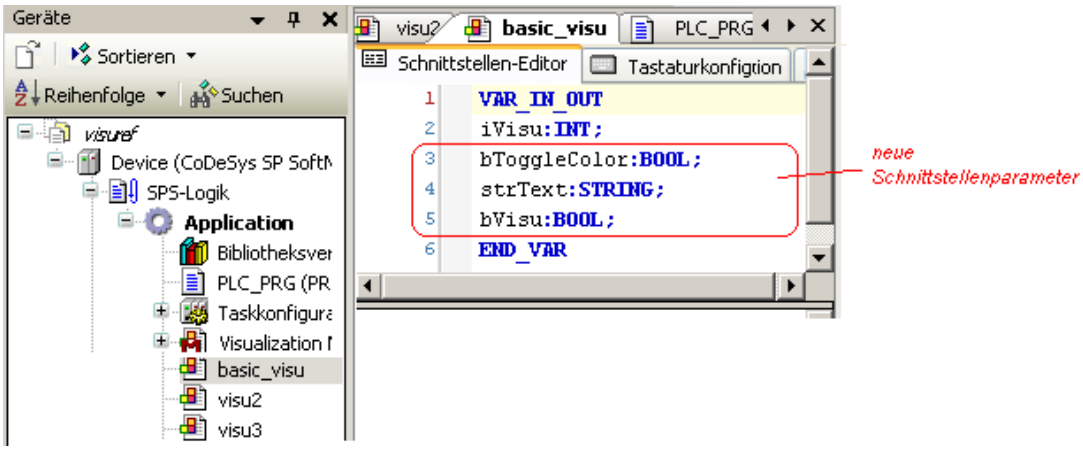


Abb. 445 Änderung im Schnittstellen-Editor

Wenn nun das Projekt beispielsweise übersetzt oder gespeichert wird, erscheint der Dialog für die Konfiguration der neuen Parameter:

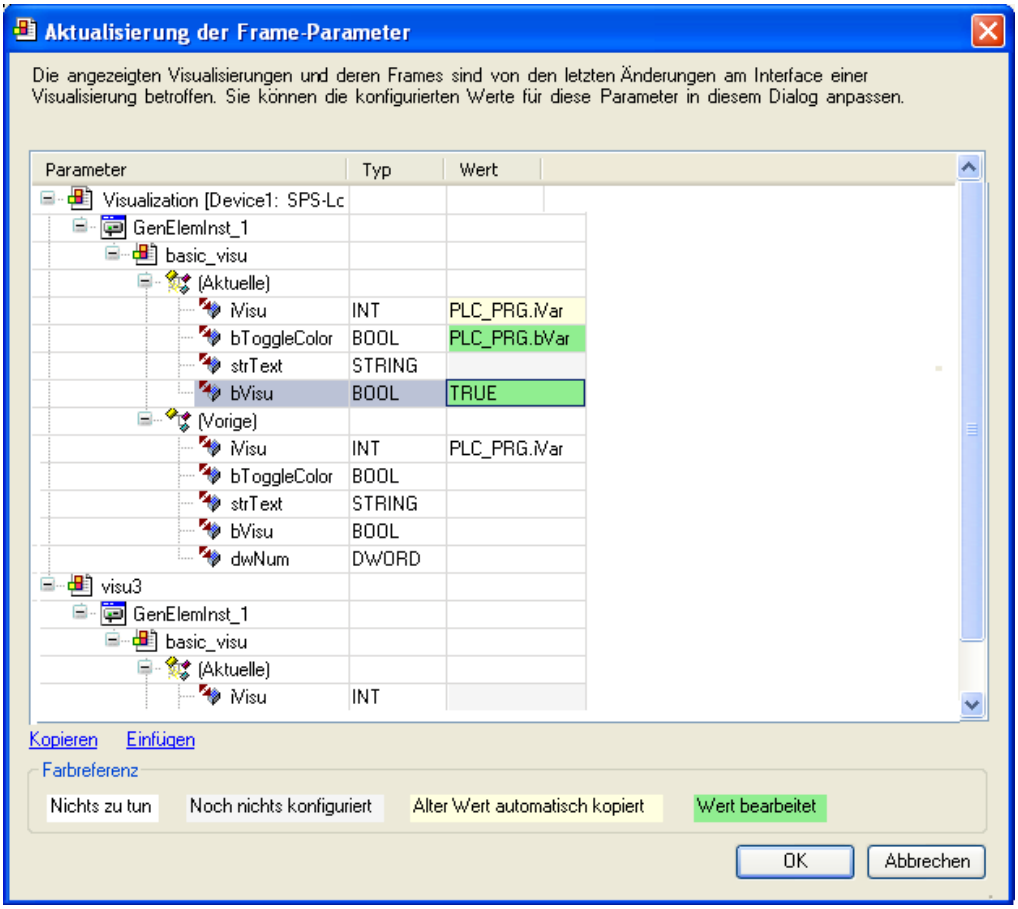


Abb. 446 Dialog Aktualisierung der Frame-Parameter

Für die Visualisierungen, die die geänderte *basic_visu* referenzieren, also *visu2* und *visu3*, werden jeweils die aktuellen und die vorigen Werte der Schnittstellenparameter dargestellt. Der aktuelle Wert kann hier neu konfiguriert werden. Die farbliche Hinterlegung der Wert-Felder gibt eine optische Unterstützung, wo noch Eingaben erforderlich sind bzw. bereits vorgenommen wurden (siehe auch Legende 'Farbreferenz' unterhalb der Parametertabelle):

beige hinterlegt: Dieser Parameter wurde automatisch aus der bisherigen Konfiguration übernommen (siehe Beispiel: *PLC_PRG.iVar* für *visu2*).

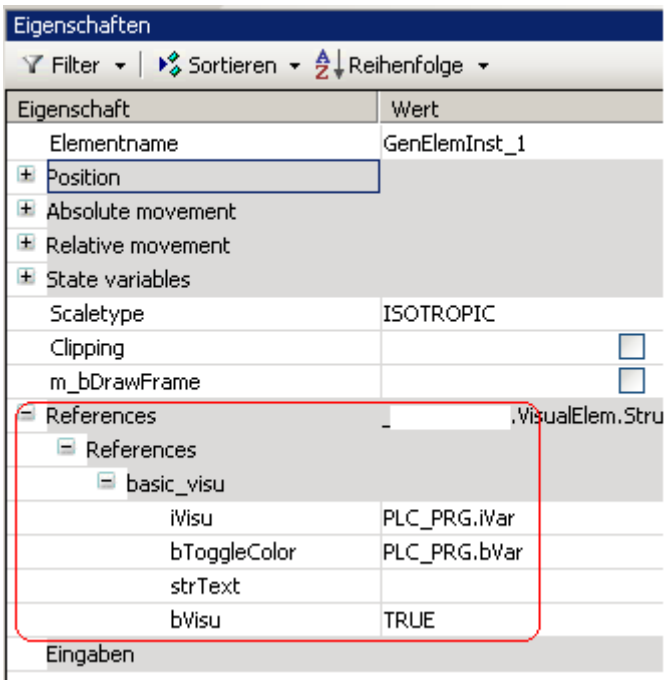
grau hinterlegt: Diesem neuen Parameter ist noch kein Wert zugewiesen (siehe Beispiel: *strText* für *visu2* und *visu3*).

grün hinterlegt: Diesem neuen Parameter ist bereits ein Wert zugewiesen (siehe Beispiel: *bToggleColor* und *bVisu* für *visu2*).

weiß hinterlegt: Hier muss nichts konfiguriert werden.

Um ein Wert-Feld zu editieren, klicken Sie mit der Maus auf das Feld und drücken die <Eingabe>-Taste. Um einen vorhandenen Wert-Eintrag zu kopieren, selektieren Sie das entsprechende Feld und drücken Schaltfläche **Kopieren,** danach selektieren Sie das Wert-Feld, in das der Wert übernommen werden soll und drücken Schaltfläche **Einfügen**.

Nach Bestätigen mit OK erscheinen die neuen Parameter-Zuweisungen im Eigenschaften-Dialog (References) des jeweiligen Frame-Elements, beispielsweise hier für den Frame in *visu2*:



Falls Sie innerhalb des referenzierten Visualisierungsbausteins (hier "basic_visu") Gebrauch machen möchten vom Instanz Namen einer übergebenen Variablen , so müssen Sie mit dem Pragma Attribut 'parameterstringof' arbeiten, welcher einen entsprechenden String zu Verfügung stellt. Sehen SIe in der Beschreibung von Attribut 'parameterstringof ein Beispiel, welches alle notwendigen Details zeigt.

13.18.7 Tastaturkonfigurations-Editor

Zusätzlich zu den Standard-Tastaturkürzeln können in diesem Teil des Visualisierungs-Editors spezielle Tastenkürzel für die Bedienung einer Visualisierung im Online-Modus definiert werden. Das heißt, eine Aktion kann einer bestimmten Taste(nkombination) zugeordnet werden. Diese Tastenkonfiguration gilt genau für die vorliegende Visualisierung. Tastenkonfigurationen, die in allen Visualisierungen der Applikation verwendbar sein sollen, sollten im Visualization Manager definiert werden.

In diesem Zusammenhang ist auch die "Hotkey" Eigenschaft eines Visualisierungselements zu beachten, die in den Elementeigenschaften (Eingabe) konfiguriert werden kann. Eine solche elementspezifische Tastenkonfiguration wird ebenfalls im Tastaturkonfigurations-Editor verwaltet. Sie kann an beiden Stellen bearbeitet werden, wobei sie jeweils auch im anderen Editor aktualisiert wird.

Der Tastaturkonfigurations-Editor - über Befehl Tastaturkonfiguration zu öffnen - ist als separates Registerblatt neben dem Schnittstellen-Editor und anderen Editoren im oberen Teil des Visualisierungs-Editors zu finden.

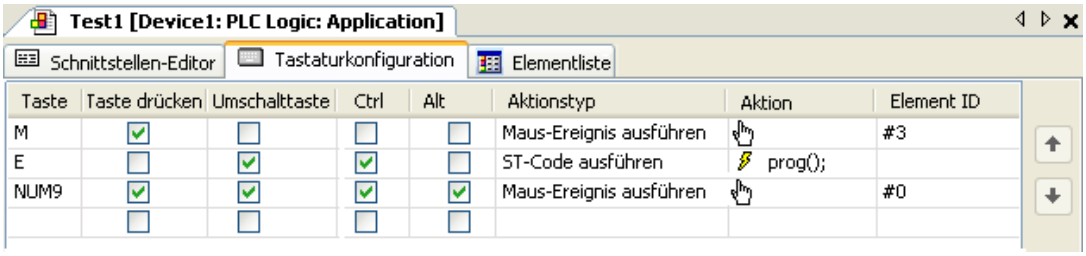


Abb. 447 Tastaturkonfiguration, Editor

In jeder Zeile des Tabelleneditors kann eine Taste oder eine Tastenkombination mit einer Aktion verknüpft werden. Siehe die folgenden Spalten:

Taste: Name der Taste. Der Tastenname kann manuell oder über eine Auswahlliste eingegeben werden, die sich bei einem Doppelklick auf die Zelle öffnet. Die Liste enthält alle Tasten, die über die **Gerätebeschreibung** definiert sind, bzw. im Falle einer Visualisierung, die keinem Gerät zugeordnet ist, alle Tasten, die von allen Geräten unterstützt werden.

Taste drücken: Wenn diese Option aktiviert ist, wird die Aktion ausgeführt sobald die Taste gedrückt wird. Ansonsten wird sie ausgeführt, wenn die Taste wieder losgelassen wird. Wenn die Aktion sowohl beim Drücken (KeyDown) als auch beim Loslassen (KeyUp) der Taste ausgeführt werden soll, müssen zwei entsprechende Definitionen für die Taste in der Tabelle vorliegen.

Umschalten, **Strg**, **Alt**: Wenn diese Option aktiviert ist, muß die Umschalt - bzw. Strg bzw. Alt Taste zusammen mit der Taste gedrückt werden, damit die Aktion ausgeführt wird.

Aktionstyp: Der Aktionstyp kann über eine Auswahlliste definiert werden, die bei einem Doppelklick auf die Zelle öffnet. Die Aktionstypen entsprechen denen, die im Eingabe-Konfigurationsdialog eines Visualisierungselements verfügbar sind.

Aktion: Genaue Konfiguration der auszuführenden Aktion; hängt vom Aktionstyp ab und entspricht der Mausaktion, wie sie in der Eingabe-Konfiguration eines Visualisierungselements verwendet werden kann.

Element ID: ID des Visualisierungselements, dem die Taste zugeordnet ist (über die "Hotkey" Eigenschaft). Eindeutige Kennung innerhalb der aktuellen Visuailsierung.

Wenn eine Taste mit mehreren Aktionen verknüpft ist, werden diese in der gleichen Reihenfolge ausgeführt, wie sie von oben nach unten hier in der Konfigurationstabelle stehen. Diese Anordnung kann geändert werden, indem eine Tastendefinition ausgewählt wird (Mausklick in die Tabellenzeile) und über die **Pfeiltasten** rechts von der Tabelle nach oben oder unten verschoben wird.

Hinweis: Es ist in der Gerätebeschreibung definiert, welche Tasten in einer Visualisierung, die auf dem Gerät läuft, unterstützt werden!

Zu beachten ist die folgende **Aufrufreihenfolge** beim Abarbeiten von Tasten-Aktionen:

1. Eventhandler der Applikation, falls aktiviert (optional); z.B. g_VisuEventManager)
2. Tastaturkonfiguration, die für alle Visualisierungen der Applikation gilt; im Visualization Manager
3. Definitionen der Standard Tastaturbedienung im Online-Modus
4. Spezielle Tastaturkonfigurationen der einzelnen Visualisierungen (verwaltet im Tastaturkonfigurations-Editor), wobei die Hauptvisualisierungen vor den in Frames liegenden beachtet werden.

13.18.8 Elementliste

Dieser Teil des Visualisierungs-Editors zeigt eine Liste aller Elemente der aktuellen Visualisierung.

Die Elementliste kann über den Befehl Elementliste (standardmäßig im Menü 'Visualisierung') geöffnet werden und ist dann als separates Registerblatt neben anderen Editoren wie z.B. dem Schnittstellen-Editor oder dem Editor für die Tastaturkonfiguration.

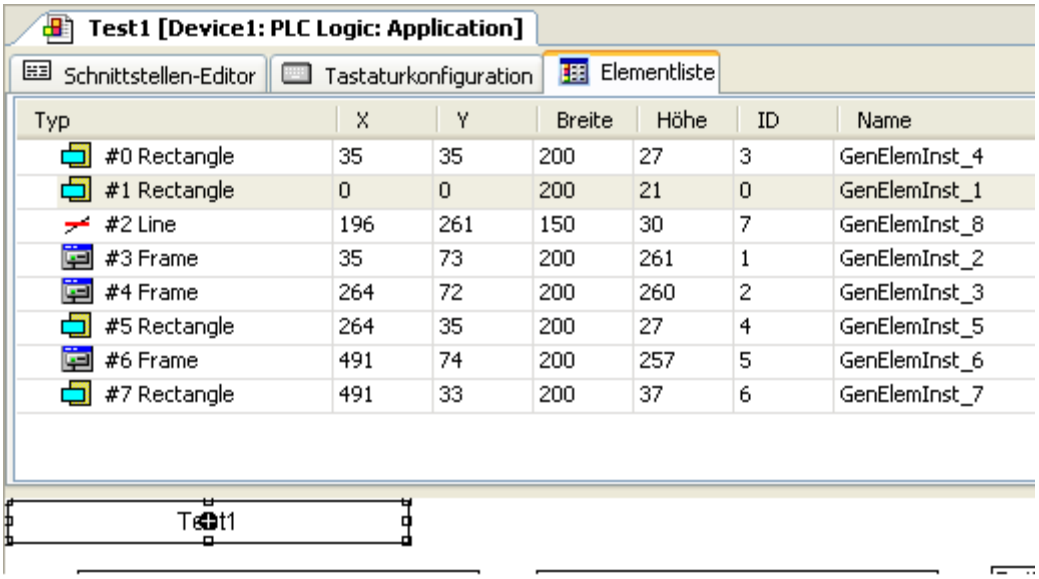


Abb. 448 Elementliste

Die Elemente werden von oben nach unten gemäß ihrer Position auf der Z-Achse der Visualisierung aufgelistet, das am weitesten im Hintergrund liegende in der ersten Zeile. Die folgenden Werte werden angezeigt, können jedoch hier nicht editiert werden:

Typ: Elementtyp und Symbol, wie in der Werkzeugbox verwendet, sowie die Elementnummer, die sich zunächst aus der Reihenfolge des Einfügens ergibt.

X, **Y**: Position der oberen linken Ecke des Elements (0,0 = obere linke Ecke des Visualisierungsbereichs)

ID: Intern zugewiesene Element-Kennung

Name. Elementname wie in den Elementeigenschaften definiert

Element(e) können durch Selektieren der entsprechenden Tabellenzeile(n) ausgewählt werden, wobei die Auswahl immer mit der im Hauptfenster des Visualisierungs-Editors synchronisiert wird. Zu beachten ist jedoch, dass **Unterelemente einer Gruppe** nur hier in der Elementliste ausgewählt werden können.

Die Position eines ausgewählten Elements auf der Z-Achse, d.h. auf der **Hintergrund-Vordergrund** Achse, kann über die folgenden Befehle verändert werden:

- Nach vorn legen
- Um Eins nach vorn legen
- Nach hinten legen
- Um Eins nach hinten legen

Die folgenden Befehle können verwendet werden, um **Bearbeitungsaktionen** in der Elementliste rückgängig zu machen oder Elemente zu entfernen:

- Undo
- Redo
- Delete

13.19 Visualisierungsmanager

Der Visualization Manager verwaltet allgemeine Einstellungen für alle Visualisierungen, die einer Applikation gehören. Einstellungsänderungen werden sofort wirksam.

Das Visualization Manager Objekt wird automatisch im Gerätebaum unterhalb einer Applikation eingefügt, sobald dieser Applikation die erste Visualisierung zugewiesen wird.

Der Hauptdialog enthält Tabulatoren für die allgemeinen Einstellungen, die Standard Tastaturkonfiguration und die Auswahl der Visualisierungen, die auf die Steuerung geladen werden.

Wenn vom Gerät unterstützt, werden im Gerätebaum unterhalb des Managers zusätzlich die Client-Objekte Target-Visualisierung und/oder Web-Visualisierung eingefügt, die die speziellen Einstellungen für Target- bzw. Web-Visualisierungen verwalten.

Zu beachten ist, dass ab V 3.3.2.0 des Programmiersystems automatisch die "Diagnose-Visualisierung" verwendet wird, wenn keine Client-Objekte unterhalb des Visualization Manager eingefügt sind. Details sehen Sie bitte unter: Visualisierung läuft im Programmiersystem.

Hinweis: Wenn in einem Projekt, das ursprünglich mit einer Version < V3.3.2 erstellt wurdet, eine feste Compiler-Version eingestellt wurde, dann verhält sich das Projekt entsprechend der Visualisierungsfunktionalitäten, die in dieser Version verfügbar waren!

In der folgenden Abbildung ist der Visualization Manager unterhalb von 'Applikation' verantwortlich für die Visualisierungen PLC_VISU, PLC_VISU_1, Vis2 und Vis3. Target-Visualisierung wird unterstützt.

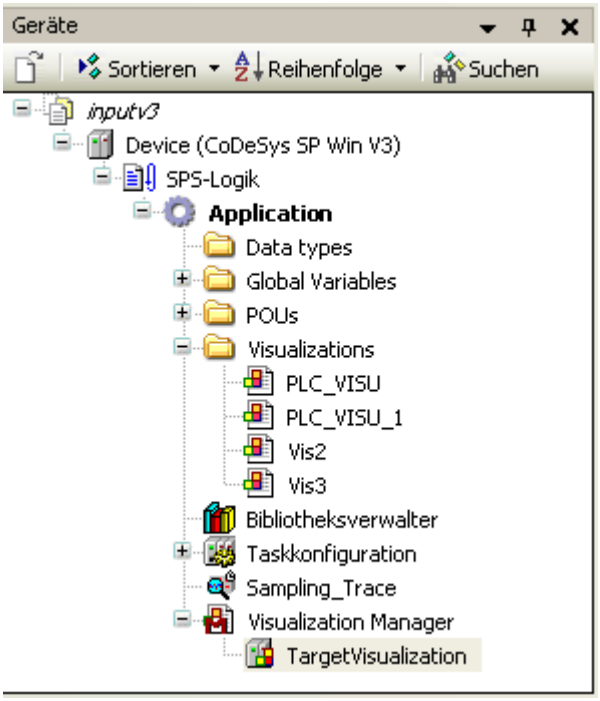


Abb. 449 Visualization Manager im Gerätebaum

Um den **Visualization Manager Editor** zu öffnen, führen Sie einen Doppelklick auf den Eintrag im Gerätebaum aus oder selektieren diesen und wählen Befehl 'Objekt bearbeiten' vom Kontextmenü. Der Editor öffnet in einem Fenster, das mit "Visualization Manager<Applikationspfad> betitelt ist und die Subdialoge Einstellungen und Standard Tastaturkonfiguration enthält:.

Einstellungen

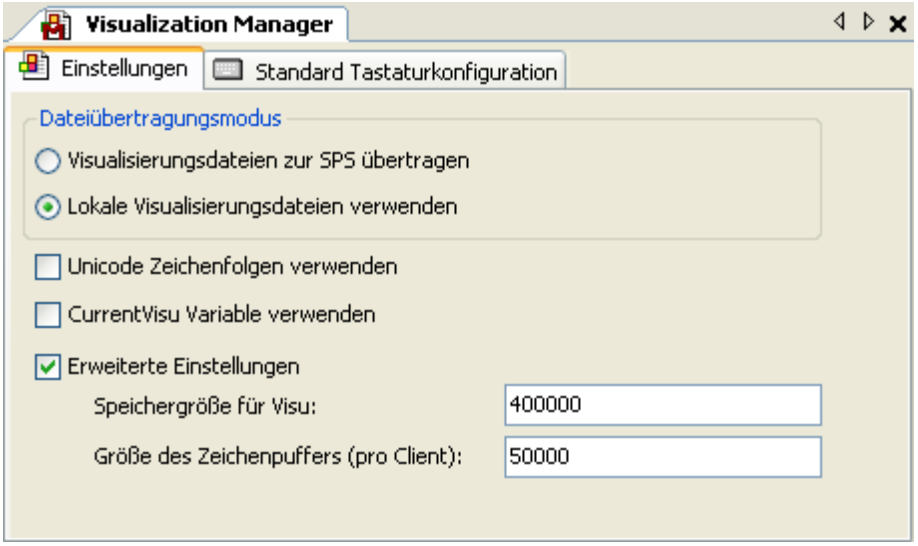


Abb. 450 Visualization Manager Editor, Einstellungen

Die Einstellungen auf diesem Registerblatt gelten für alle applikations-zugehörigen Visualisierungen:

Hinweis: Wenn die Diagnose-Visualisierung verwendet wird, sind die dafür aktuell nicht verfügbaren Einstellungen gegraut dargestellt.

Dateiübertragungsmodus:

Visualisierungsdateien zur SPS übertragen: Wenn diese Option aktiviert ist, werden beim Download der Applikation die für die Visualisierung benötigten Dateien (z.B. Bilddateien, Textlisten) vom lokalen Dateiverzeichnis (siehe nachfolgend beschriebene Option) auf die Steuerung kopiert. Dies wird beispielsweise für die Target-Visualisierung benötigt, wie auch für die Standard-Numpad- und Keypad-Visualisierungen, die in der Eingabekonfiguration eines Visualisierungselements verwendet werden.

Lokale Visualisierungsdateien verwenden: Die für die Visualisierung benötigten Bild- und Textlisten-Dateien werden aus den lokalen Dateiverzeichnissen gelesen, die in den Projekteinstellungen, Kategorie Visualisierung definiert sind.

Unicode Zeichenfolgen verwenden: Wenn diese Option aktiviert ist, werden alle Zeichenfolgen, die in der Visualisierung verwendet werden, in Unicode-Format verarbeitet.

CurrentVisu Variable verwenden: Bei Aktivierung dieser Option wird der Wert der Variable "CurrentVisu" berücksichtigt. Indem man dieser Variablen als Wert den Namen der entsprechenden Visualisierung zuweist, kann man kontrollieren, welche Visualisierung innerhalb des Programms angezeigt wird.

Erweiterte Einstellungen: Aktivieren Sie diese Option, wenn folgende Einstellungen geändert werden müssen (was für eine Standardanwendung nicht erforderlich sein sollte):

Speichergröße für Visu: Speicherplatz in Bytes, der für die Visualisierung reserviert ist; Default: 400000

Größe des Zeichenpuffers (pro Client): Speicherpuffer in Bytes, der für Zeichenaktionen auf dem Client-System reserviert ist; Standard: 50000

Standard Tastaturkonfiguration

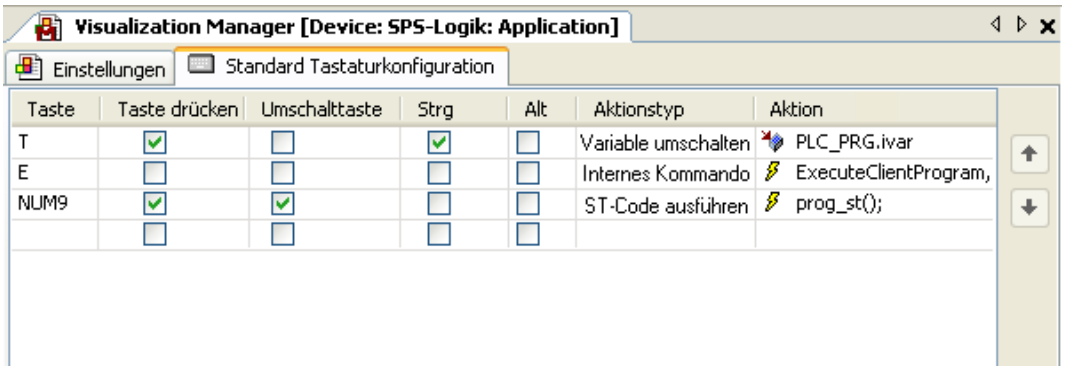


Abb. 451 Visualization Manager Editor, Standard Tastaturkonfiguration

Dieses Registerblatt des Visualization Manager Dialogs enthält die Tastaturkonfiguration, die für alle der Applikation zugehörigen Visualisierungen gilt. Das heißt hier definierte Tasten(kombinationen) können - wenn das jeweilige Gerät sie unterstützt - für Benutzereingaben in der Visualisierung im Online-Betrieb verwendet werden. Der Konfigurator ist genau so zu bedienen wie der, der im Visualisierungs-Editor für eine spezielle Visualisierung verwendet wird. Siehe somit die Beschreibung des Tastaturkonfiguration-Editors.

Zusätzlich gelten immer geräteunabhängig einige bestimmte Standard Tastaturkürzel für das Navigieren in einer Visualisierung.

Visualisierungen

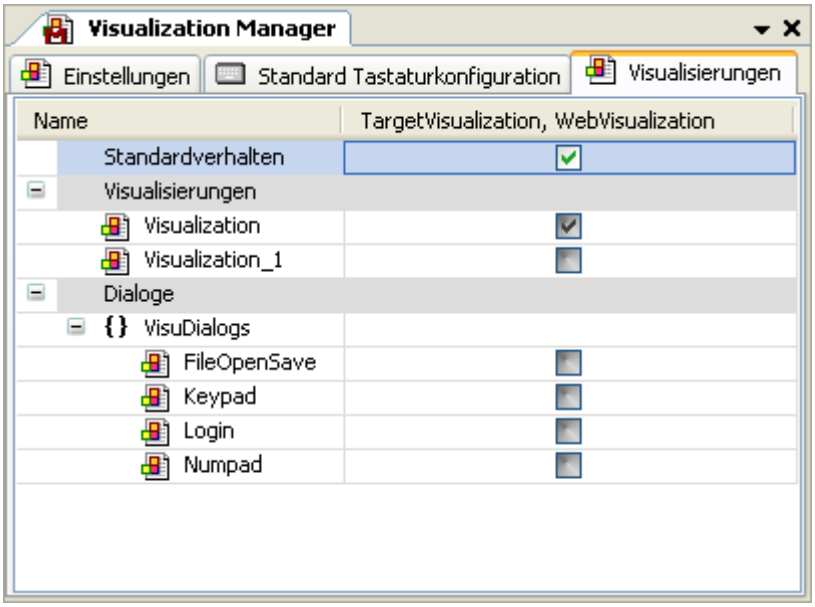


Abb. 452 Visualization Manager Editor, Visualisierungen

Dieses Registerblatt des Visualization Manager Dialogs zeigt die gerade verfügbaren Visualisierungen an. Standardmäßig werden diejenigen, die unter der aktuellen Applikation hängen oder die direkt von anderen referenziert werden, automatisch auf das jeweilige Zielsystem geladen. Wenn die Option **Standardverhalten** aktiviert ist, sind genau diese in der Tabelle mit einem Häkchen in der jeweiligen Spalte "TargetVisualization" oder "WebVisualization" versehen. Durch explizites Setzen eines solchen Häkchens können hier dem Download-Paket weitere Visualisierungen hinzugefügt werden, die standardmäßig nicht enthalten sind, weil sie nur indirekt verwendet/referenziert werden (z.B. durch eine Variable in der Applikation).

Hinweis: Diese Funktionalität ersetzt die vorher mögliche Verwendung von "Visualisierungsreferenz" Objekten. Dieses Objekt kann jetzt nicht mehr neu eingefügt werden, bereits eingefügte funktionieren allerdings weiterhin.

13.19.1 Target-Visualisierung

Das Objekt "Target-Visualisierung", das unterhalb des Visualization Manager Objekts eingehängt ist, enthält die Einstellungen für die Target-Visualisierung. Sie können im Editorfenster bearbeitet werden, das nach einem Doppelklick auf das Objekt öffnet.

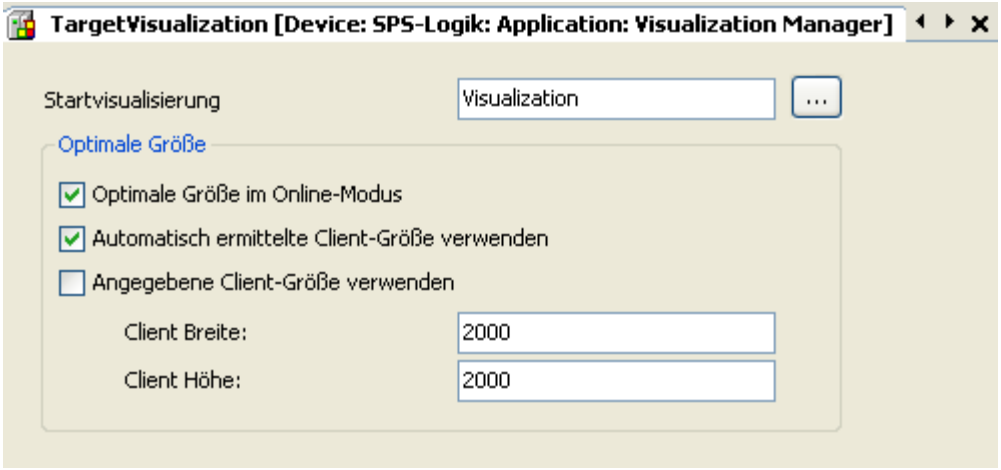


Abb. 453 Target-Visualisierung, Dialog

Beachten Sie die Voraussetzung für das Betreiben einer Target-Visualisierung, die auf der Hilfeseite zur Target-Visualisierung beschrieben sind.

Startvisualisierung: Name der Visualisierung, die automatisch als Target-Visualisierung angezeigt werden soll, wenn das Zielsystem gestartet wird. Standardmäßig ist hier "Visualisierung" bereits eingetragen. Zur Auswahl einer anderen Visualisierung kann die Eingabehilfe verwendet werden.

Optimale Größe:

Optimale Größe im Online-Modus: Wenn diese Option nicht aktiviert ist, wird die Target-Visualisierung in genau der Größe angezeigt, die über die Visualisierungs-Objekteigenschaften definiert ist. Wenn die Option aktiviert ist, wird die Darstellungsgröße gemäß folgender Einstellungen angepaßt:

Automatisch berechnete Client-Größe verwenden: Die Target-Visualisierung füllt den Client-Bildschirm aus.

Angegebene Client-Größe verwenden: Die Target-Visualisierung füllt den durch folgende Maße bestimmte Bildschirmbereich:

Client Breite: Breite in Pixel

Client Höhe: Höhe in Pixel

13.19.2 Web-Visualisierung

Das **Web-Visualisierung** Objekt unterhalb des Visualization Manager Objekts enthält die Einstellungen für die Web-Visualisierung. Sie können im Editorfenster, das bei einem Doppelklick auf das Objekt geöffnet wird, bearbeitet werden:

Beachten Sie zum Prinzip und den Voraussetzungen für das Betreiben einer Web-Visualisierung die Hilfeseite zur Web-Visualisierung.

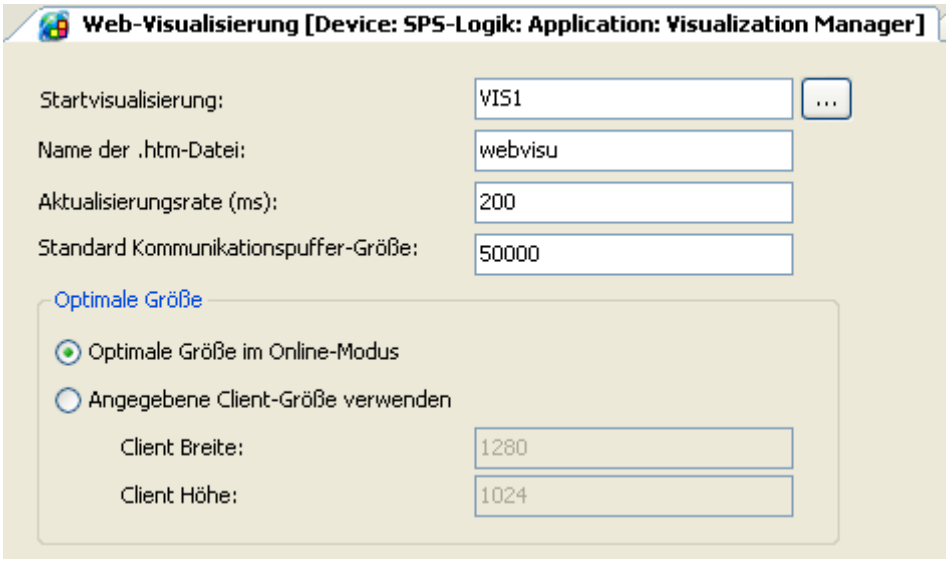


Abb. 454 Web-Visualisierung, Dialog

Startvisualisierung: Name der Visualisierung, die automatisch angezeigt werden soll, wenn die Web-Visualisierung gestartet wird. Standardmäßig ist hier "Visualisierung" bereits eingetragen. Zur Auswahl einer anderen Visualisierung kann die Eingabehilfe verwendet werden.

Name der .htm Datei: Name der Basis-HTML-Seite der Visualisierung, welche dann auch als Adresse im Web-Browser einzugeben ist. Beispiel: http://localhost:8080/webvisu.htm

Aktualisierungsrate (ms): Aktualisierungsrate innerhalb des Web-Browsers in Millisekunden.

Standard Kommunikationspuffer-Größe: Kommunikationspuffergröße in Bytes, gibt den maximal verfügbaren Speicher für die Übertragung der Daten zwischen Web-Client und Web-Browser an.

Optimale Größe:

Optimale Größe im Online-Modus verwenden: Wenn diese Option aktiviert ist, wird die Anzeigegröße der Web-Visualisierung der Größe des Browser-Fensters angepaßt.

Angegebene Client-Größe verwenden: Wenn diese Option aktiviert ist, wird die Anzeigegröße der Web-Visualisierung durch die folgenden Einstellungen definiert:

Client Höhe: Höhe in Pixel

Client Breite: Höhe in Pixel

13.20 Überwachen-Editor

Das Plug-In 'Watch List' stellt die Funktionalität der Überwachungslisten bereit.

Eine Überwachungsliste ist eine benutzerdefinierte Zusammenstellung von Projektvariablen, die im Überwachungsfenster zum Zweck des Monitorings ihrer Werte in Tabellenform dargestellt werden. Auch ein Schreiben und Forcen der Variablen ist im Überwachungsfenster möglich.

Ein Editor-Fenster für das Definieren einer Überwachungsliste kann mit dem Befehl Überwachen <n> im Untermenü 'Überwachen" geöffnet werden, das standardmäßig im Menü 'Ansicht' zu finden ist. Standardmäßig sind vier Überwachungsfenster vorbereitet: **Überwachen 1**, **Überwachen 2**, **Überwachen 3**, **Überwachen 4**. Die Ansicht **Alle Forces betrachten** wird automatisch mit allen gerade "geforcten" Werten innerhalb der aktiven Applikation gefüllt.

13.20.1 Überwachungsliste anlegen

Um in einem der Überwachungsfenster eine Variable für die Überwachungsliste **Überwachen <n>** zu definieren, öffnen Sie ein Eingabefeld in Spalte **Ausdruck**. Dazu muss das Feld ausgewählt und die <Leertaste> gedrückt werden. Geben Sie den kompletten Pfad der gewünschten Variablen ein, die Eingabehilfe steht über Schaltfläche ... bereit.

Syntax für einen Überwachungsausdruck:

<Gerätename>.<Applikationsname>.<Objektname>.<Variablenname>,

Beispiel: "Dev1.App1.PLC_PRG.ivar" (wie in der unten gezeigten Abbildung)

Durch ein Symbol vor dem Ausdruck wird angezeigt ob es sich um eine Eingabevariable, Ausgabevariable oder "normale" Variable handelt.

Wenn die Eingabe des Ausdrucks abgeschlossen ist, erscheint in Spalte **Datentyp** automatisch der Typ der Variable. Wenn der Deklaration der Variable ein Kommentar hinzugefügt wurde, wird dieser in Spalte **Kommentar** angezeigt.

Die Spalte **Wert** zeigt dann im Online-Modus den aktuellen Variablenwert an.

Um für eine Variable einen Wert vorzubereiten, klicken Sie in das zugehörige Feld der Spalte **Vorbereiteter Wert** und tragen dort den gewünschten Wert ein.

Im Fall einer boolschen Variablen kann zdurch Drücken der **Return** oder **Space** Taste wischen den boolschen Werten hin- und hergeschaltet werden, gemäß folgender Reihenfolge: Ist der bisherige Wert TRUE, so erfolgen Wechsel gemäß FALSE -> TRUE -> kein Eintrag; andernfalls, falls der bisherige Wert also FALSE ist, erfolgen Wechsel gemäß der Reihenfolge TRUE -> FALSE -> kein EIntrag.

Geben Sie so alle gewünschten Variablen in den Zeilen des Fensters ein. Siehe unten eine Abbildung mit dem Beispiel einer Überwachungsliste im Offline-Modus: Die Liste enthält Ausdrücke aus den Objekten PLC_PRG und Prog_St. Beachten Sie im Fall einer Strukturvariable, wie hier die Funktionsblockinstanz, dass die einzelnen Komponenten automatisch in weiteren Zeilen eingefügt werden, wenn der Instanzname eingetragen wird (siehe im Beispiel: "Dev1.App1.PLC_PRG.fbinst"). Die Komponenten können durch Betätigen des Plus- bzw. Minuszeichens (Klammerung) aus- oder eingeblendet werden.

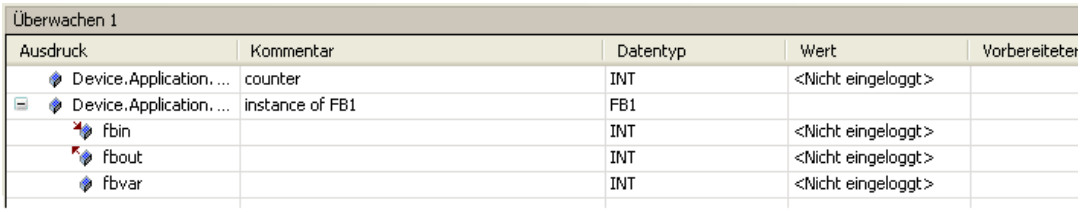


Abb. 455 Beispiel, Überwachungsfenster im Offline-Betrieb

Im Online-Modus kann die Liste für das Monitoring der Variablenwerte verwendet werden.

13.20.2 Überwachungsliste im Online-Modus

Monitoring:

Im Online-Modus wird der Wert der Variablen bzw. des Ausdrucks in der Spalte **Wert** einer Überwachungsliste angezeigt.

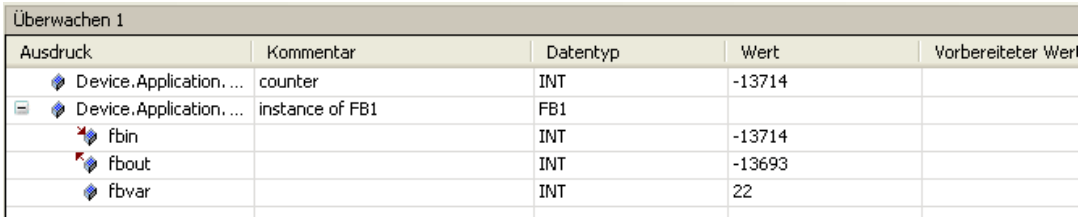


Abb. 456 Beispiel, Überwachungsliste im Online-Modus

Werte schreiben oder forcen:

In der Spalte **Vorbereiteter Wert** kann ein Wert eingegeben werden, der auf die Variable in der Steuerung geschrieben werden kann bzw. mit dem sie "geforced" werden kann. Sehen Sie hierzu die Beschreibungen zum Schreiben und Forcen von Variablen, das auch in anderen Monitoring-Ansichten wie z.B. im Deklarationseditor möglich ist.

Alle Forces betrachten:

Dies ist eine spezielle Überwachen-Ansicht, die zur Laufzeit automatisch mit allen gerade geforcten Werten der aktiven Applikation gefüllt wird. Es werden jeweils Ausdruck, Datentyp, Wert und Vorbereiteter Wert angezeigt, wie in den 'Überwachen <n>' Listen.

Unter der Schaltfläche **Force aufheben...** stehen folgende Kommandos zum Aufheben geforcter Werte zur Verfügung:

- **Den Force aller selektierten Symbole aufheben, ohne den Wert zu ändern.**
- **Den Force aller selektierten Symbole aufheben und den ursprünglichen Wert wiederherstellen**

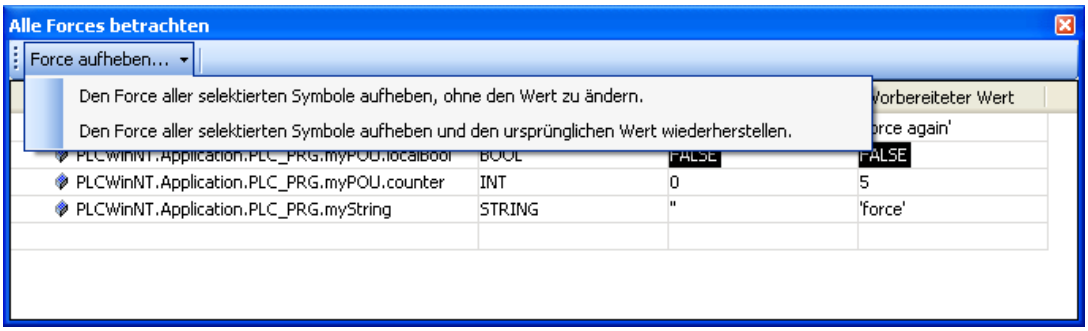


Abb. 457 Beispiel - Alle Forces betrachten

14 Geräte-Editoren

14.1 Generischer Geräte-Editor

Der Geräte-Editor stellt Dialoge für die Konfiguration eines Geräts bereit, das im Geräte-Fenster des Projekts verwaltet wird.

Er kann mit einem Doppelklick auf das Geräte-Objekt im Geräte-Fenster geöffnet werden, oder mit Befehl 'Objekt bearbeiten', wenn das Geräte-Objekt ausgewählt ist.

Beachten Sie den Dialog der Geräteeditor-Optionen für generelle Einstellungen. Beispielsweise kann dort die Anzeige von generischen Dialogen aus- oder eingeschaltet werden.

Standardmäßig wird der **Geräte-Dialog** mit einem Doppelklick auf das Objekt, oder mit dem Befehl 'Objekt bearbeiten' geöffnet, wenn das Geräte-Objekt im Gerätebaum selektiert ist.

Der Hauptdialog trägt als Titel den Gerätenamen, z.B.

, und abhängig vom Gerätetyp enthält er eine Kombination aus den folgenden Unterdialogen:

- Kommunikationseinstellungen: Konfiguration der Verbindung zwischen Programmiersystem und einem programmierbaren Gerät (SPS). Nicht verfügbar bei reinen I/O-Geräten.
- Konfiguration: Anzeige bzw. Konfiguration der Geräteparameter.
- Applikationen: Liste der Applikationen auf der Steuerung.
- Dateien: Konfiguration einer Dateiübertragung zwischen Host und Steuerung.
- Log: Anzeige der SPS Log-Datei.
- SPS-Einstellungen: Konfiguration der Applikation bezüglich der Handhabung der E/As, des Verhaltens der Ein- und Ausgänge im Stop-Status; Buszyklus-Optionen
- I/O Abbild: Abbilden der Ein- und Ausgangskanäle des Geräts auf Projekt-(Applikations-) variablen.
- Benutzer und Gruppen: Benutzerverwaltung bezüglich Zugriffe auf die Steuerung zur Laufzeit.
- Zugriffsrechte: Rechte für Zugriffe auf Objekte und Dateien auf dem Gerät.
- Status: Gerätespezifische Status- und Diagnosemeldungen.
- Information: Allgemeine Information zum Gerät (Name, Hersteller, Version etc.)
- Bus-spezische Konfigurationsdialoge, z.B.:
  - DP Profibus Konfiguration
  - Ethercat Konfiguration

Hinweis: Die Dialoge 'Kommunikationseinstellungen' und 'Applikationen' betreffen die Kommunikation mit der Steuerung und werden vom Plug-In 'Device Communication Editor' bereitgestellt.

Kommunikationseinstellungen

Dieser Unterdialog des Geräte-Dialogs (Geräte-Editor) dient der Konfiguration der Parameter für die Kommunikation zwischen Gerät und Programmiersystem.

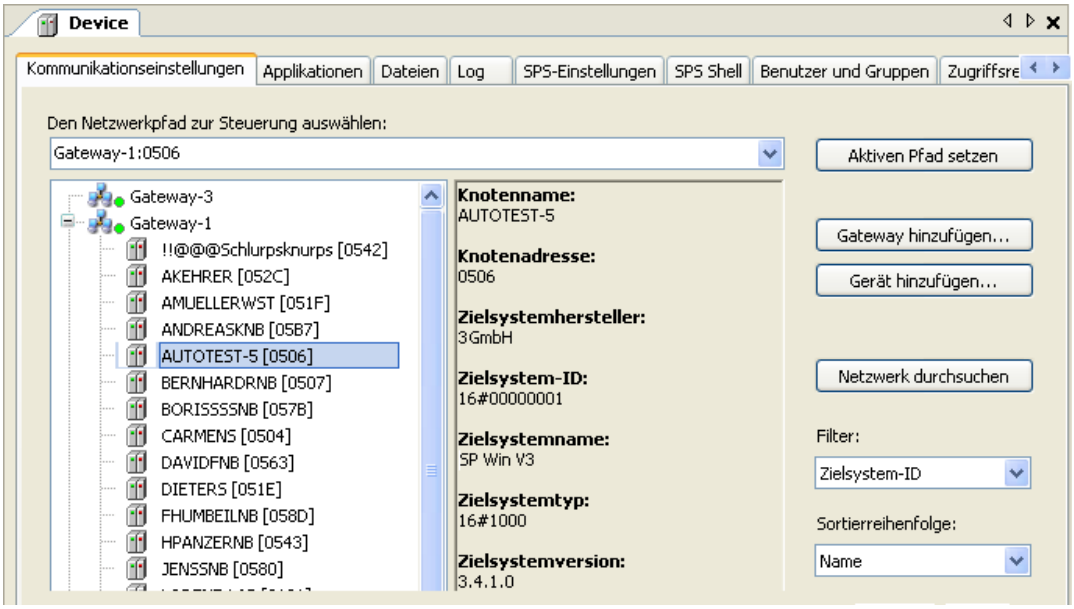


Abb. 458 Geräte-Dialog, Kommunikationseinstellungen

Der Dialog enthält ein zweigeteiltes Fenster, das im linken Teil in einer Baumstruktur die aktuell konfigurierten Gateway-Kanäle und im rechten Teil die zugehörigen Daten und Informationen anzeigt.

Wenn das erste Projekt auf dem lokalen System erstellt wird, liegt auf jeden Fall bereits ein Eintrag im Baum vor: Der lokale Gateway (der beim Systemstart automatisch gestartet wird):

Die Gateway-Einstellungen werden im rechten Fensterteil angezeigt (siehe auch unten: 'Gateway'-Dialog): Beispiel:

Knotenname: "Gateway-1"
Port: 1217
IP-Adresse: 127.0.0.1
Treiber: TCP/IP

Solange der Gateway korrekt läuft, wird der Kreis im Symbol vor dem Eintrag grün gefüllt dargestellt , ansonsten ein roter . Der Kreis bleibt grau, wenn der Gateway noch nicht verbunden wurde. Manche Kommunikationsprotokolle erlauben kein regelmäßiges Prüfen des Gateways, so dass der Status gar nicht angezeigt werden.

Eingerückt unterhalb des Gateway-Eintrags (Auf- und Zuklappen über die +/- Zeichen) sieht man die Einträge aller Geräte, die über den Gateway erreichbar sind: Man beachte, dass diese Einträge lokal auf dem System und nicht im Projekt gespeichert werden. Einträge mit einer Target-ID, die sich von der gerade im Projekt konfigurierten unterscheiden, sind in grauer Schrift dargestellt. Eine aktualisierte Liste erhält man über die Schaltfläche 'Netzwerk durchsuchen' (s.u.).

Die Geräte-Einträge im Baum bestehen jeweils aus einem Symbol gefolgt von "<**Knotenname**> [**Knotenadresse**]" und im rechten Fensterteil werden zusätzlich **Zielsystem-ID**, **Zielsystemname**, **Zielsystemtyp**, **Zielsystemhersteller** und **Zielsystemversion** angezeigt.

Die Darstellung des Gateway- und Gerätebaums kann mit Hilfe von Filtern und Sortierfunktionen kontrolliert werden. Siehe hierzu die Auswahlmöglichkeiten im rechten Teil des Dialogs:

- **Filter:** Die Liste kann auf die Anzeige nur solcher Geräte reduziert werden, die die gleiche **Zielsystem-ID** haben wie das aktuell im Projekt konfigurierte Gerät.
- **Sortieren:** Die Liste kann nach **Name** bzw. **Knotenadresse** in alphabetischer bzw. aufsteigender Reihenfolge sortiert werden.

Im Feld '**Den Netzwerkpfad zur Steuerung auswählen**' muss der gewünschte Gateway-Kanal eingetragen sein. Dies erfolgt automatisch durch Auswählen des Kanals im Baum.

Für das Bearbeiten der Kommunikationseinstellungen stehen die folgenden Schaltflächen bzw. Befehle aus dem Kontextmenü zur Verfügung:

Aktiven Pfad setzen	Dieser Befehl setzt den gerade ausgewählten Kommunikationskanal als den aktiven. Siehe "Aktiven Pfad setzen. Ein Doppelklick auf den Eintrag im Kanälebaum bewirkt dasselbe.
Gateway hinzufügen...	Dieser Befehl öffnet den "Gateway" Dialog, wo Sie einen Gateway definieren können, der zur aktuellen Konfiguration hinzugefügt werden soll. Siehe "Gateway hinzufügen"
Gerät hinzufügen...	Dieser Befehl öffnet den Dialog "Gerät hinzufügen", wo manuell ein Gerät definiert werden kann, das unter dem gerade ausgewählten Gateway-Eintrag eingefügt werden soll (Beachten Sie dazu auch die 'Netzwerk durchsuchen' Funktionalität). Siehe "Gerät hinzufügen..."
Gateway bearbeiten...	Dieser Befehl öffnet den "Gateway" Dialog zum Bearbeiten der Einstellungen für den gerade ausgewählten Gateway. Siehe "Gateway bearbeiten"
Ausgewähltes Gerät löschen	Dieser Befehl entfernt das ausgewählte Gerät von der Geräteliste. Siehe "Ausgewähltes Gerät löschen..."
Suche nach Gerät über Name	Dieser Befehl (Kategorie Gerätekommunikation) dient dazu, das Netzwerk nach Geräten zu durchsuchen, die denselben Namen tragen wie im Dialog Kommunikationseinstellungen in der Geräteliste angegeben (Beachtung der Groß/Kleinschreibung!). Wenn welche gefunden werden, wird jedes davon mit seiner Knotenadresse der Liste hinzugefügt, d. h. der Eintrag wird mit Name + Knotenadresse dargestellt. Siehe "Suche nach Gerät über Name "
Suche nach Gerät über Adresse	Dieser Befehl dient dazu, das Netzwerk nach Geräten zu durchsuchen, die denselben Kontenadresse haben wie im Dialog Kommunikationseinstellungen in der Geräteliste angegeben . Wenn welche gefunden werden, wird jedes davon mit seinem Namen der Liste hinzugefügt, d. h. der Eintrag wird mit Name + Knotenadresse dargestellt. Siehe "Suche nach Gerät über Adresse "
Zum lokalen Gateway verbinden	Dieser Befehl öffnet einen Dialog zur Konfiguration eines lokalen Gateway und bietet daher eine Alternative zur manuellen Bearbeitung der Datei "Gateway.cfg". Siehe "Zum lokalen Gateway verbinden"
Netzwerk durchsuchen	Dieser Befehl startet eine Suche für verfügbare Geräte im lokalen Netzwerk. Der Konfigurationsbaum des betreffenden Gateways wird entsprechend aktualisiert. Siehe "Netzwerk durchsuchen"

Sicherer Online-Betrieb: Wenn diese Option aktiviert ist, wird der Benutzer aus Sicherheitsgründen beim Aufrufen der folgenden Online-Befehle jeweils zu einer Bestätigung aufgefordert: Werte forcen, Werte schreiben, Mehrfaches Laden, Forceliste für <Applikation> aufheben, Einzelzyklus, Start <Applikation>, Stop <Applikation>.

Netzwerkpfad nicht im Projekt speichern: Wenn diese Option aktiviert ist, wird die aktuelle Netzwerkpfad-Definition nicht im Projekt gespeichert, sondern in den lokalen Optionen-Einstellungen auf Ihrem Computer. Somit wird die Einstellung wieder hergestellt, wenn das Projekt auf dem gleichen Computer wieder geöffnet wird, muss aber neu gesetzt werden, wenn es auf einem anderen verwendet wird.

Konfiguration

Dieser Unterdialog ist nur dann Geräte-Editor verfügbar, wenn die Option 'Generische Konfigurationseditoren anzeigen' aktiviert ist. Er zeigt die gerätespezifischen Parameter an und, wenn dies durch die Gerätebeschreibung so definiert ist, können diese hier auch bearbeitet werden.

Beispiel für ein Profibus Modul, alle Parameter sind nicht editierbar:

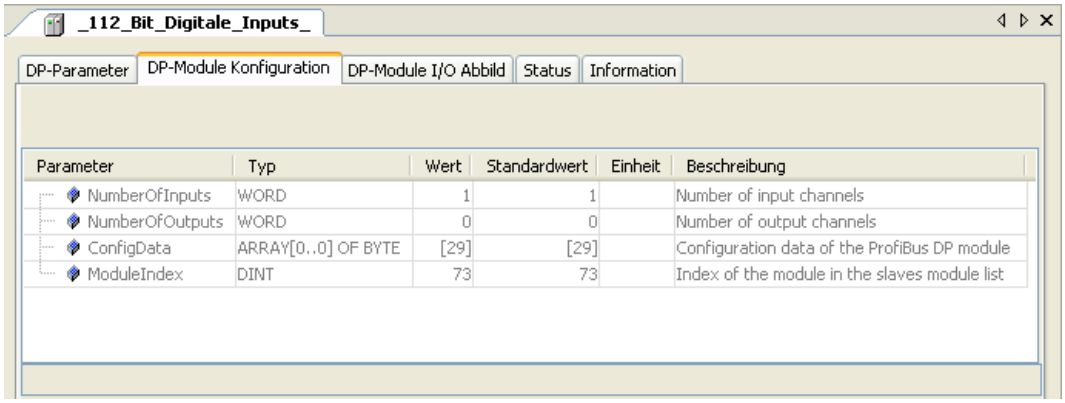


Abb. 459 Geräte-Dialog, Konfiguration

Die folgenden Spalten dienen der Beschreibung eines Parameters:

Parameter: Parametername, nicht editierbar.

Typ: Datentyp des Parameters, nicht editierbar.

Wert: Zunächst wird hier der Standardwert des Parameters angezeigt, direkt oder als Angabe des entsprechenden Symbolnamens. Wenn der Parameter bearbeitet werden kann (dies hängt von der Gerätebeschreibung ab, nicht-editierbare Parameter werden hellgrau dargestellt), kann mit einem Doppelklick auf das Tabellenfeld (oder mit der <Leertaste> bei einem bereits selektierten Feld) ein **Eingabefeld** oder eine **Auswahlliste** geöffnet werden, über die der Wert geändert werden kann. Wenn es sich bei dem Wert um eine Dateiangabe handelt, öffnet sich der Standarddialog zum Auswählen einer Datei.

Standardwert: Von der Gerätebeschreibung definierter Standardwert, nicht editierbar.

Einheit: Maßeinheit für den Wert, z.B. "ms" für Millisekunden, nicht editierbar.

Beschreibung: Kurzbeschreibung zum Parameter, nicht editierbar.

Applikationen

Dieser Unterdialog des Geräte-Dialogs (Geräte-Editor) dient dazu, die Applikationen, die gerade auf der Steuerung liegen, anzuzeigen und ggfs. zu löschen.

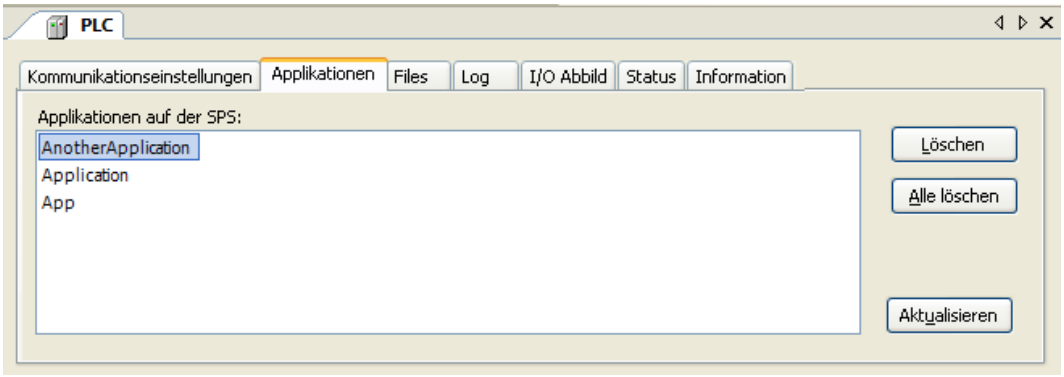


Abb. 460 Geräte-Dialog, Applikation

Applikationen auf der SPS: Liste der Applikationen, die beim letzten "Scan" der Steuerung (durch 'Aktualisieren') dort gefunden wurden. Wenn noch kein Scan durchgeführt wurde, bzw. dieser nicht möglich ist, weil noch kein Gateway für eine Verbindung konfiguriert wurde, wird dafür ein Hinweis ausgegeben.

Aktualisieren: Die Steuerung wird nach Applikationen durchsucht, die Liste wird entsprechend aktualisiert.

Löschen resp. **Alle löschen**: Die gerade in der Liste selektierte Applikation bzw. alle Applikationen werden von der Steuerung gelöscht.

Wenn eine Applikation in die Steuerung geladen wird, wird vorher folgendes geprüft:

- Die Liste der Applikationen auf der Steuerung wird mit den im Projekt verfügbaren verglichen. Bei Nicht-Übereinstimmung erscheinen die entsprechenden Dialoge, entweder zum Laden der noch nicht auf der Steuerung vorhandenen Applikation oder zum Löschen von anderen Applikationen auf der Steuerung.
- Für "extern implementierte" Bausteine in der zu ladenden Applikation wird geprüft, ob diese auch auf der Steuerung verfügbar sind. Wenn dies nicht der Fall ist, wird eine entsprechende Meldung ("unaufgelöste Referenz(en)") in einer Meldungsbox und im Meldungsfenster ausgegeben.
- Die Parameter (Variablen) der Bausteine in der zu ladenden Applikation werden mit denen der gleichnamigen Bausteine in der bereits auf der Steuerung vorliegenden Applikation verglichen (Signaturen-Prüfung). Bei Unstimmigkeiten wird eine entsprechende Meldung ("ungültige Signatur(en)") in einer Meldungsbox und im Meldungsfenster ausgegeben.

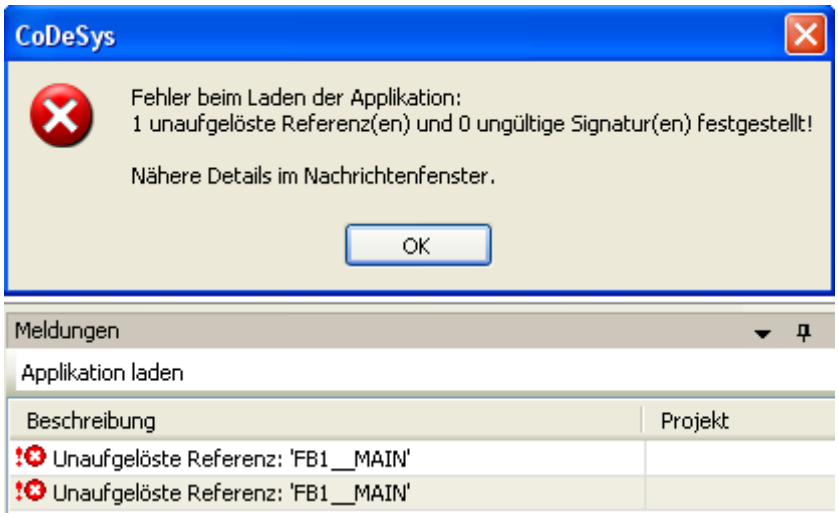


Abb. 461 Beispiel, Meldung zu geprüften Referenzen und Signaturen

Wenn für dasselbe Gerät mehrere Applikationen bereitstehen, beachten Sie, dass für das I/O-Mapping des Geräts im Dialog I/O Abbild definiert wird, welche der Applikationen beim Mapping berücksichtigt wird.

Siehe auch Applikation für generelle Informationen zur Handhabung von Applikationsprojekten.

Dateien

Dieser Unterdialog des Geräte-Dialogs (Geräte-Editor) dient dem Datei-Transfer zwischen »PLC Designer« (Host) und Steuerung. Das heißt, eine oder mehrere Dateien aus dem lokalen Dateisystem können in das Dateienverzeichnis des gerade verbundenen Zielsystems kopiert werden und umgekehrt.

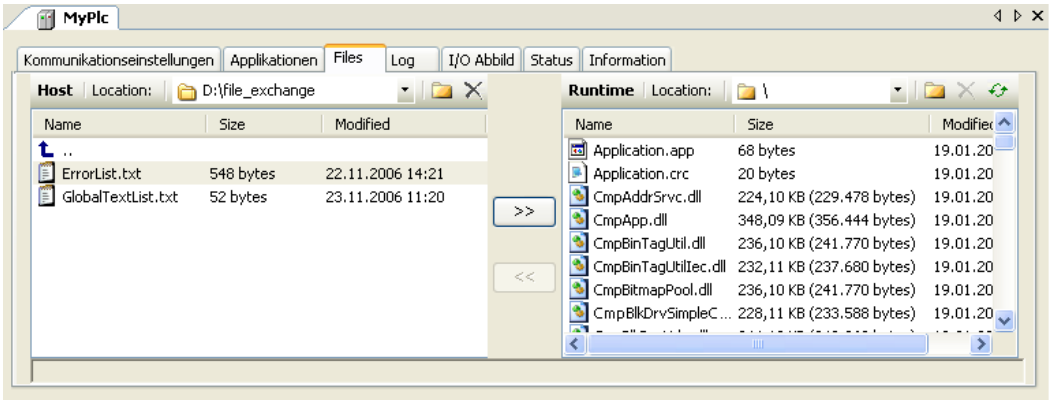


Abb. 462 Geräte-Dialog, Dateien

Im linken Teil des Dialogs wird das Dateisystem des **Hosts** abgebildet, im rechten das des **Zielsystems (Laufzeit)**. Oberhalb der Listen stehen jeweils folgende Funktionen zur Verfügung:

Mit Schaltfläche kann die Dateienliste aktualisiert werden.

Mit Schaltfläche kann ein neuer Ordner angelegt werden, in den die Datei(en) kopiert werden soll(en).

Mit Schaltfläche werden die gerade in der Liste ausgewählten Dateien bzw. Ordner gelöscht.

Im Feld **Ort** muss der aktuell für den Dateitransfer zu berücksichtigende Ordner des jeweiligen Dateisystems eingetragen sein. Dies geschieht entweder über die Auswahlliste direkt an diesem Feld oder über Browsen unten im Dateisystembaum.

Die Dateien, die kopiert werden sollen, werden dann im Dateisystembaum ausgewählt. Mehrfachauswahl ist möglich, ebenso kann ein Ordner angewählt werden, wenn alle enthaltenen Dateien übertragen werden sollen.

Um die ausgewählten Dateien in das auf der Host- bzw. im Zielsystem-Seite eingestellte Verzeichnis zu transferieren, wird die Schaltfläche **>>** bzw. **<<** betätigt. "Transferieren" bedeutet in diesem Fall "Kopieren". Das heißt, wenn eine Datei im Ziel-Verzeichnis noch nicht vorhanden ist, wird sie dort neu angelegt, wenn sie bereits vorliegt, wird sie überschrieben, außer sie ist schreibgeschützt (Dann wird eine entsprechende Meldung ausgegeben).

Log

Dieser Unterdialog des Geräte-Dialogs (Geräte-Editor) dient dem Anzeigen des "Logbuchs" der Steuerung, d.h. dem Anzeigen von Ereignissen, die auf dem Zielsystem aufgezeichnet wurden. Dies betrifft:

- Ereignisse beim Systemstart und -Beenden (geladene Komponenten mit Version)
- Applikations-Download und Laden des Bootprojekts
- Kundenspezifische Einträge
- Log-Einträge von I/O-Treibern
- Log-Einträge vom DataServer

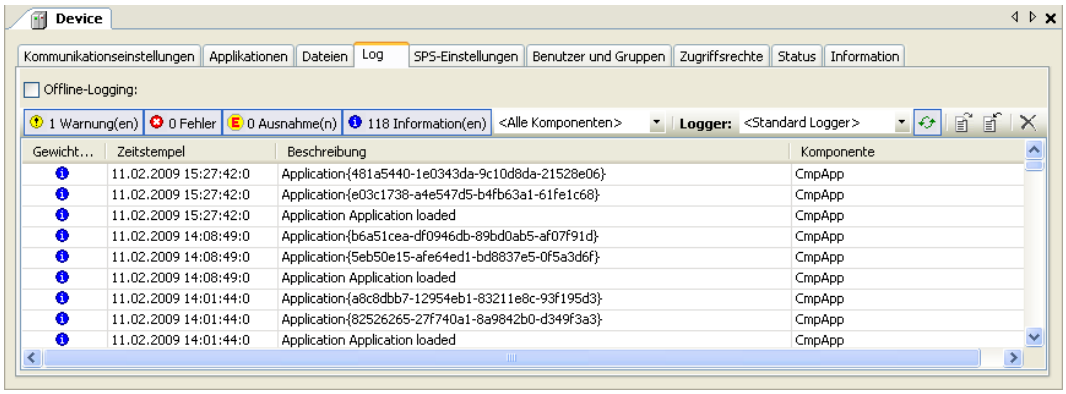


Abb. 463 Geräte-Editor, Log

Ein Logbucheintrag wird mit den folgenden Informationen dargestellt:

Gewichtung: Für die Schwere des Ereignisses gibt es vier Kategorien: Warnungen, Fehler, Ausnahmefehler, Information. Die Darstellung jeder Kategorie kann ein- oder ausgeblendet werden, indem die entsprechende Schaltfläche in der Leiste oberhalb der Liste betätigt wird. In der Schaltfläche wird jeweils die Anzahl der Log-Einträge der betreffenden Kategorie angezeigt.

Zeitstenpel: Datum und Uhrzeit, z.B. "12.01.2007 09:48"

Beschreibung: Beschreibung des Ereignisses, z.B. "Import function failed of <CmpFileTransfer>"

Komponente: Name der betroffenen Komponente.

Liste mit Komponentennamen: Hier kann aus einer Auswahlliste eine einzelne Komponente ausgesucht werden, so dass nur Logbucheinträge, die diese betreffen, angezeigt werden. Die Standardeinstellung ist 'Alle Komponenten'.

Logger: Die Auswahlliste enthält die verfügbaren Aufzeichnungen. Die Standardeinstellung ist der vom Zielsystem vorgegebene "<Default Logger>", derzeit identisch mit "PlcLog" für das »PLC Designer« SP Laufzeitsystem.

noch nicht verfügbar: Die Log-Liste wird automatisch aktualisiert.

Derzeit kann die Liste mit Hilfe der Schaltfläche aktualisiert werden.

Der Inhalt der Liste kann in eine xml-Datei exportiert oder aus einer solchen importiert werden. Für den Export verwenden Sie Schaltfläche , um den Standarddialog zum Speichern einer Datei zu öffnen. Der Dateifilter ist auf "xml-files (*.xml)" eingestellt. Die Log-Liste wird mit dem angegebenen Dateinamen mit Erweiterung ".xml" im gewählten Verzeichnis abgelegt. Um die Log-Einträge, die in einer xml-Datei möglicherweise auf diese Weise gespeichert wurden, anzuzeigen, verwenden Sie die Schaltfläche . Der Standarddialog zum Öffnen einer Datei erscheint. Auch hier ist der Filter auf "xml-files (*.xml)" gesetzt. Wählen Sie die gewünschte Log-Datei und deren Einträge werden in einem separaten Fenster dargestellt.

Um die aktuelle Log-Liste zu leeren, d.h. alle angezeigten Einträge zu löschen, verwenden Sie Schaltfläche .

Wenn Option **Offline-Logging** aktiviert ist, werden auch Aktionen, die sich nicht auf die Verbindung mit der Steuerung beziehen, aufgezeichnet. Dies ist momentan jedoch nur für die Safety-Version des Programmiersystems realisiert.

SPS-Einstellungen

Dieser Unterdialog des Geräte-Editors dient der Konfiguration einiger Grundeinstellungen für das SPS-Gerät.

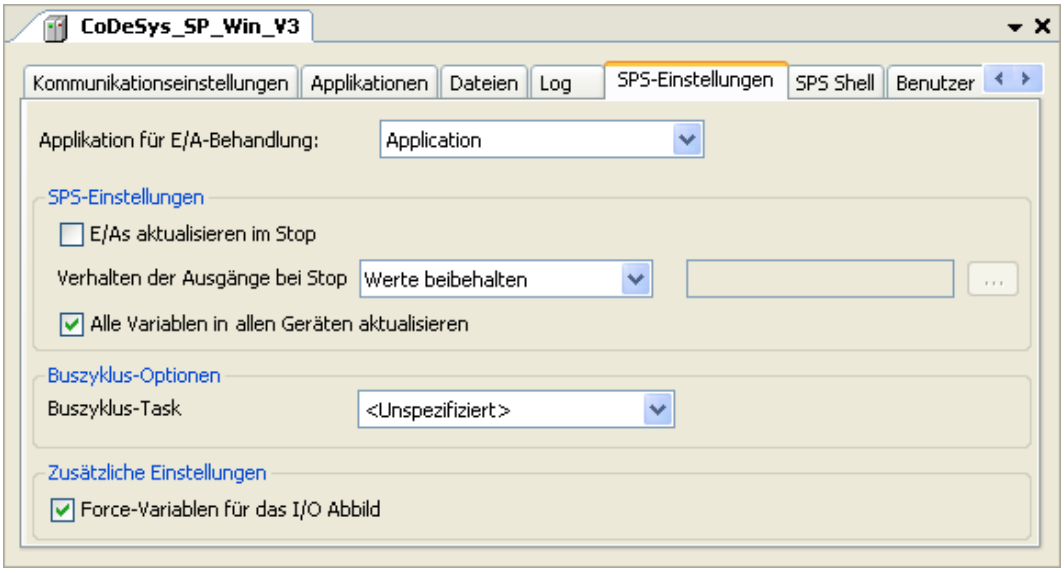


Abb. 464 Geräte-Dialog, SPS-Einstellungen

Applikation für E/A-Behandlung: Hier wird diejenige der dem Gerät zugewiesenen Applikationen angegeben, die für die E/A-Behandlung (I/O-Behandlung) zuständig sein soll. Zunächst ist die Standard-Applikation, die in einem Standard-Projekt automatisch eingefügt ist, eingetragen, aber von der Auswahlliste kann eine beliebige andere Applikation genommen werden.

SPS-Einstellungen:

E/As aktualisieren im Stop: Wenn diese Option aktiviert ist (Default), werden die Werte der Ein- und Ausgangskanäle auch aktualisiert, wenn die SPS in den Stop-Status geht. Stellt der Watchdog eine Fehlfunktion fest, so werden die Ausgänge auf die vordefinierten Standardwerte gesetzt.

Verhalten der Ausgänge bei Stop: Die Auswahlliste bietet folgende Möglichkeiten, wie die Werte der Ausgangskanäle behandelt werden sollen, wenn die Steuerung in den Stop-Status geht:

Werte beibehalten: Die augenblicklichen Werte werden beibehalten.

Alle Ausgänge auf Standardwert: Die Standardwerte, die aus dem I/O Abbild resultieren, werden zugewiesen.

Programm ausführen: Die Behandlung der Ausgangswerte kann über ein im Projekt enthaltenes Programm gesteuert werden. Der Name dieses Programms kann hier angegeben werden und es wird dann ausgeführt, wenn die Steuerung in Stop-Status geht. Mit der Schaltfläche [...] kann die Eingabehilfe geöffnet werden, um die Programmauswahl zu erleichtern.

Alle Variablen in allen Geräten aktualisieren: WEnn diese Option aktiviert ist, werden für alle Geräte der aktuellen Steuerungskonfiguration in jedem Zyklus der Buszyklus-Task alle E/A-Variablen aktualisiert. Dies entspricht der Option Variablen immer aktualisieren, die für jedes Gerät auch separat im "I/O Abbild"-Dialog gesetzt werden kann.

Buszyklus-Optionen:

Buszyklus-Task: Die Auswahlliste bietet alle Tasks an, die in der Taskkonfiguration der aktiven Applikation definiert sind. (z.B. "MainTask", "EtherCAT Master" etc.). Die Standardeinstellung, die in der Gerätebeschreibung vorgegeben ist, ist bereits automatisch gesetzt. Es kann eine andere gewählt werden, dann **muss jedoch unbeding folgendes bedacht werden:**

> **Hinweis:** Bevor auf Einstellung "<unspezifiziert>"gewechselt wird, sollte man sich über den Effekt klar sein. "<unspezifiziert>" bedeutet, dass das Standardverhalten, wie es in der Gerätebeschreibung definiert ist, wirksam wird. Deshalb muss die Beschreibung diesbezüglich geprüft werden. Als Standard kann das Verwenden der Task mit der kürzesten Zykluszeit definiert sein, genauso gut aber auch das Verwenden derjenigen mit der längsten Zykluszeit!

Zusätzliche Einstellungen:

Force-Variablen für das I/O Abbild: Diese Einstellung ist nur verfügbar, wenn sie vom Gerät unterstützt wird. Wenn die Option aktiviert ist, werden für jeden I/O-Kanal, der im Dialog "I/O Abbild" auf eine Variable abgebildet ("gemapped") ist, beim Übersetzen der Applikation zwei globale Variablen erzeugt. Diese können für das Forcen des Eingangs-bzw. Ausgangswerts an diesem Kanal verwendet werden, beispielsweise über eine HMI Visualisierung. Siehe E/A-Abbild für mehr Informationen.

I/O Abbild

Dieser Unterdialog des Geräte-Dialogs (Geräte-Editor) ist betitelt mit "<Gerätetyp> I/O Abbild" (z.B. "Profibus DP I/O Abbild"). Er dient dem Konfigurieren eines I/O Abbilds der Steuerung ("I/O mapping"). Das heißt, den Eingangs-, Ausgangs- und Speicheradressen des Steuerungsgeräts werden Projektvariablen zugewiesen, die von der Applikation verwendet werden.

Die Applikation, die die I/O-Behandlung übernehmen soll, muss im Dialog SPS-Einstellungen festgelegt werden.

Siehe im folgenden:

- Allgemeine Information zum I/O Abbild
- Arbeiten im Dialog
- Implizite Variablen zum Forcen von I/Os

Allgemeines

Ob ein I/O Abbild für das aktuelle Geräteobjekt konfiguriert werden kann, hängt vom Gerät ab; ggfs. beschränkt sich der Dialog auf die Anzeige einer implizit angelegten Geräteinstanz, s.u.: "IEC Objekte".

Beachten Sie generell das folgende für das Abbilden von I/Os auf Variablen:

- Auf Variablen, die auf einen Eingang gelegt sind, kann nicht schreibend zugegriffen werden.
- Statt den I/O Abbild Dialog zu verwenden kann eine I/O-Adresse auch über eine "AT-Deklaration" mit einer Variablen verknüpft werden. Beachten Sie hierzu: 1. Eine AT-Deklaration kann nur bei lokalen oder globalen Variablen angewendet werden, nicht jedoch bei Ein- oder Ausgabevariablen von Bausteinen. 2. Die Möglichkeit, "Force-Variablen" für I/Os zu erzeugen (siehe unten: Implizite Variablen zum Forcen von I/Os) steht für "AT declarations" nicht zur Verfügung. 3. Wenn eine AT-Deklaration bei Struktur- oder Funktionsblockvariablen verwendet wird, werden alle Instanzen auf die gleiche Speicherstelle zugreifen, was den "statischen Variablen" in klassischen Programmiersprachen wie z.B. C entspricht.
- Das Speicher-Layout von Strukturen ist wird vom Zielsystem vorgegeben.
- Für jede Variable, die einem I/O-Kanal im "I/O Abbild" Dialog zugewiesen wird, können während des Übersetzens der Applikation "Force-Variablen" erzeugt werden. Diese können beispielsweise bei der Inbetriebnahme einer Maschine oder Anlage für das Forcen eines Wertes auf dem Eingang bzw. Ausgang verwendet werden, z.B. über eine Visualisierung (HMI). Siehe unten: Implizite Variablen für das Forcen von I/Os.

Arbeiten im Dialog

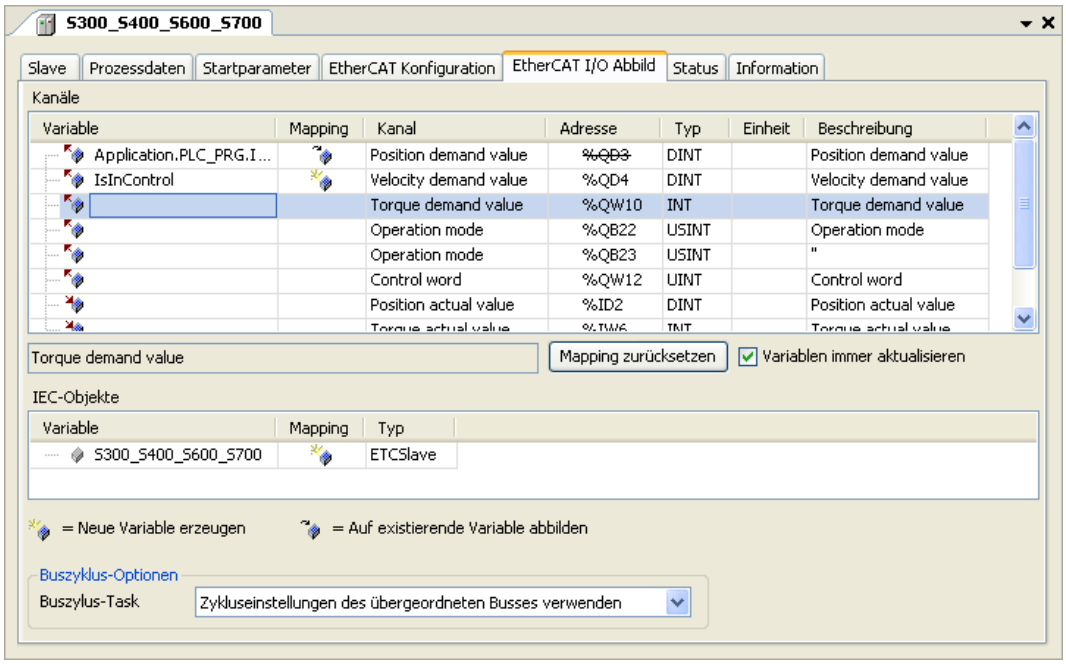


Abb. 465 Geräte-Dialog, I/O Abbild

Kanäle:

Folgende Informationen aus der Gerätebeschreibung werden in der Tabelle für jeden Geräteein- bzw. -ausgangskanal angezeigt:

Kanal: Symbolischer Name des Kanals.

Adresse: Adresse des Kanals, beispielsweise "%IW0".

Der aktuell angezeigte Adresswert kann hier im Dialog geändert und fixiert werden. Dies kann nützlich sein um die Adressierung an eine vorgegebene Maschinenkonfiguration anzupassen bzw. um den Adresswert auch dann beizubehalten, wenn sich die Anordnung der Module ändert (was standardmäßig zu einer automatischen Anpassung der Adresswerte führen würde).

Zu beachten ist, dass nur die Adresse des "gesamten" Ein- oder Ausgangs zu diesem Zweck bearbeitet werden kann, nicht die seiner Unterelemente, z.B. Bitkanäle. Also kann, wenn ein Eingang oder Ausgang hier in der Mapping-Tabelle mit einem Unterbaum dargestellt ist, nur das Adressfeld des obersten Eintrags editiert werden (siehe die Abbildung unten: nur das Adressfeld in der ersten Zeile kann geöffnet werden).

Zum Fixieren eines Adresswerts muss der Eintrag in der "Adresse"-Spalte ausgewählt und die <Leertaste> gedrückt werden, um das Eingabefeld zu öffnen. Dann kann entweder der Wert geändert oder belassen werden und mit der <Eingabe> Taste das Eingabefeld wieder geschlossen. Daraufhin wird das Adressfeld mit einem Symbol versehen, was anzeigt, dass der Adresswert fixiert ist. Wenn der Wert verändert wurde, werden die nachfolgenden Adressen (bis zur nächsten fixierten) automatisch entsprechend angepasst. Siehe folgendes Beispiel der Änderung und Fixierung eines Adresswertes:

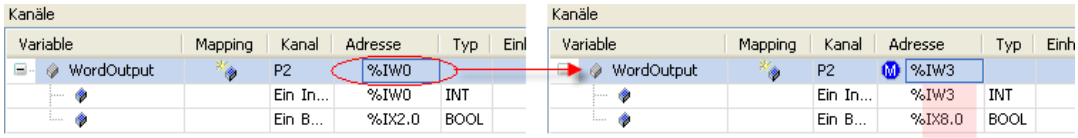


Abb. 466 Beispiel, manuell geänderte Adresse

Um die manuelle Änderung bzw. Fixierung rückgängig zu machen, öffnen Sie erneut das Eingabefeld des Adresswerts, löschen den Adresseintrag und drücken die <Eingabe> Taste. Die Adresse und die betroffenen nachfolgenden Adressen werden auf die Werte zurückgesetzt, die sie vor der Änderung hatten und das M-Symbol wird entfernt.

Typ: Datentyp des Kanals, z.B. "BOOL". Wenn es sich nicht um einen Standard-Datentypen handelt, sondern um eine in der Gerätebeschreibung definierte Struktur oder ein Bitfeld, wird er nur angezeigt, wenn er Teil der IEC-Norm ist und als IEC-Datentyp in der Gerätebeschreibung gekennzeichnet ist. Ansonsten bleibt die Tabellenzelle leer.

Einheit: Einheit für den Parameterwert, z.B. "ms" für Millisekunden

Beschreibung: Kurze Beschreibung des Parameters.

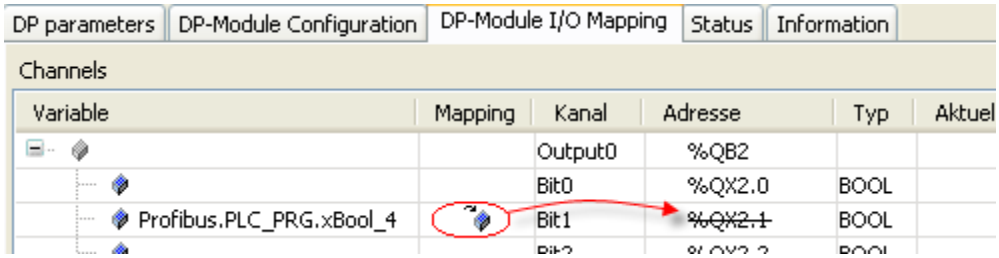
Aktueller Wert: Aktueller Wert des auf dem Kanal anliegenden Parameters, wird im Online-Modus angezeigt.

Das I/O-Abbild (Mapping) wird nun konfiguriert, indem den Geräteein- und ausgangskanälen in der Spalte 'Variable' die entsprechenden Projektvariablen zugewiesen werden.

Der **Typ** eines Kanals ist in Spalte **Variable** durch ein Symbol angezeigt: für Eingang, für Ausgang. In dieser Tabellenzelle wird der Name bzw. Pfad der Variablen eingegeben, auf die der Kanal abgebildet werden soll. Dabei kann entweder eine bereits existierende Projektvariable angegeben, oder eine neue definiert werden, die dann automatisch intern als globale Variable deklariert wird.

Beim Mapping von **strukturierten Variablen** verhindert der Editor, dass sowohl die Strukturvariable (beispielsweise auf %QB0) als auch einzelne Strukturelemente (beispielsweise dann auf %QB0.1 und QB0.2 etc.) eingetragen werden können. Das heißt: Wenn es einen Haupteintrag mit einem Unterbaum von Bitkanal-Einträgen in der Mapping-Tabelle gibt (wie in der Abbildung unten zu sehen), dann kann entweder in der Zeile des Haupteintrags eine Variable eingegeben werden, oder in denen der Unterelemente (Bitkanäle), aber nicht in beiden.

- Beim **Abbilden auf eine bestehende Variable** muss der komplette Pfad eingegeben werden, d.h. "<Applikationsname>.<Bausteinname>.<Variablenname>"; Beispiel: "app1.plc_prg.ivar". Dazu wird am besten die Eingabehilfe über Schaltfläche geöffnet. In der **Mapping** Spalte erscheint das Symbol ("Auf existierende Variable abbilden") und der Adresswert wird **durchgestrichen** dargestellt. Dies heißt nicht, dass es diese Speicheradresse nicht mehr gibt. Sie wird nur nicht direkt verwendet, weil der Wert einer bestehenden Variablen auf einem anderen Speicherplatz verwaltet wird und es sollte, speziell bei Ausgängen, auf keinen Fall eine andere bereits existierende Variable auf diese Adresse gespeichert werden ("%Qxx" im I/O-Abbild), um Uneindeutigkeiten beim Schreiben der Werte zu vermeiden.
  Sehen Sie im folgenden Beispiel ein Ausgangs-Mapping auf eine bestehende Variable xBool_4:



- Wenn **eine neue Variable definiert** wird, muss nur ein Variablenname eingegeben werden; Beispiel: "bVar1". In diesem Fall wird in der **Mapping** Spalte das Symbol eingefügt und die Variable automatisch **intern als globale Variable deklariert.** Ab diesem Zeitpunkt ist die Variable global innerhalb der Applikation verfügbar.

Hinweis: Monitoring: Wenn eine Strukturvariable auf das "Root"-Element (oberstes im Baum der betreffenden Adresse im Mapping-Dialog) einer Adresse abgebildet wird, wird im Online-Betrieb in dieser Zeile kein Wert angezeigt. Wenn jedoch beispielsweise eine DWORD-Variable auf der Adresse abgebildet wird, werden sowohl in der "Root"-Zeile als auch in den darunter eingerückten Bit-Kanal-Zeilen die jeweiligen Werte gemonitored. Grundsätzlich bleibt das Feld in der "Root"-Zeile immer dann leer, wenn sich der Wert aus mehreren Unterelementen zusammensetzen würde.

Hinweis: Abbilden "zu großer" Datentypen: Wenn eine Variable eines Datentyps, der größer ist als ein Byte, auf eine Byte-Adresse abgebildet wird, wird dort der Wert der Variablen auf Byte-Größe abgeschnitten! Für das Monitoring des Variablenwerts im Mapping-Dialog bedeutet dies: Im "Root"-Element der Adresse wird der Wert angezeigt, den die Variable im Projekt aktuell hat. In den Bit-Elementen darunter werden nacheinander die einzelnen aktuellen Bit-Werte des Bytes dargestellt, was aber gegebenenfalls nicht für den gesamten Variablenwert ausreicht.

Mapping zurücksetzen: Über diese Schaltfläche können die Mapping-Einstellungen auf die durch die Gerätebeschreibungsdatei definierten Standardwerte zurückgesetzt werden.

Variablen immer aktualisieren: Wenn diese Option aktiviert ist, werden alle Variablen in jedem Zyklus der Buszyklus-Task aktualisiert, egal ob sie verwendet werden und ob sie auf einen Ein- oder Ausgangskanal abgebildet sind.

Hinweis: Alternativ kann eine Adresse auch im Programmier-Code gelesen oder geschrieben werden. Beispiel in ST:

```
Var := IB100;
QX12.3 := TRUE;
```

In Anbetracht möglicher Änderungen der Gerätekonfiguration wird empfohlen, solche Zuweisungen im Gerätekonfigurationsdialog vorzunehmen.

IEC Objekte:

Dieser Teil des Dialogs erscheint nur, wenn implizit eine **Instanz** des Geräteobjekts angelegt wird, auf die in der Applikation zugegriffen werden kann (beispielsweise, um einen Bus neu zu starten oder Informationen abzufragen). Ob eine solche Instanz zur Verfügung steht und wie sie verwendet werden kann, hängt vom Gerätetyp ab und wird auf den entsprechenden Hilfeseiten für die spezielle Gerätekonfiguration beschrieben.

Buszyklus-Optionen:

Diese Einstellungen sind für Geräte verfügbar, bei denen vor und nach dem Lesen von Ein- oder Ausgängen zyklische Aufrufe erfolgen. Sie erlauben eine bus-spezifische Task zu definieren.

Standardmäßig gilt die Buszyklus-Einstellung des übergeordneten Busgeräts (**Zykluseinstellungen des übergeordneten Busses verwenden**), das heißt der Gerätebaum wird nach oben auf die nächste gültige Buszyklus-Taskdefinition hin durchsucht.

Um die spezifische Buszyklus-Task zuzuweisen, wählen Sie die gewünschte aus der Auswahlliste. Die Liste enthält alle Tasks, die aktuell in der Taskkonfiguration der Applikation definiert sind.

Implizite Variablen zum Forcen von I/Os

Während der Inbetriebnahme einer Anlage oder Maschine kann es erforderlich sein, I/Os zu forcen, beispielsweise über eine HMI Visualisierung. Zu diesem Zwecke können spezielle "Force-Variablen" erzeugt werden, für jeden I/O-Kanal, der im "I/O Abbild" Dialog des Geräteeditors auf eine Variable abgebildet ist.

Also Voraussetzung muss die Einstellung **Variablen für das I/O Abbild erzeugen** im Dialog SPS-Einstellungen aktiviert sein. Dann werden bei jedem **Übersetzen** der Applikation für jeden I/O-Kanal zwei Variablen angelegt, gemäß der folgenden Syntax, wobei Leerzeichen im Kanalnamen mit Unterstrichen ersetzt werden:

- "Gerätename“_“Kanalname“_“IECAdresse“_“Force" vom Typ BOOL, für das Aktivieren und Deaktiveren des Forcens
- "Gerätename“_“Kanalname“_“IECAdresse“_“Value" vom Datentyp des Kanals, für das Definieren des Wertes, der am Kanal geforced werden soll

Diese Variablen sind dann in der Eingabehilfe in Kategorie "Variablen", "IoConfig_Globals_Force_Variables" verfügbar. Sie können in Programmierobjekten, in Visualisierungen, Symbolkonfiguration etc. im Programmiersystem verwendet werden.

Eine steigende Flanke an der "Force"-Variable aktiviert das Forcen des jeweiligen Eingangs/Ausgangs mit dem Wert, der durch die "Value"-Variable gegeben ist. Eine fallende Flanke deaktiviert das Forcen. Das Deaktiviern durch Rücksetzen der "Force"-Variable auf FALSE ist Voraussetzung dafür, dass ein neuer Wert geforced werden kann!

Man beachte die weiter unten aufgelisteten Einschränkungen!

Beispiel

Wenn das Mapping wie oben in Abbildung "Geräte-Dialog, I/O Abbild, Beispiel" gezeigt vorliegt, dann werden beim Übersetzungslauf (F11) der Applikation die folgenden Variablen erzeugt und in der Eingabehilfe bereitgestellt:

- Digitax_ST_Control_word_QW0_Force : BOOL;
- Digitax_ST_Control_word_QW0_Value : UINT;
- Digitax_ST_Target_position_QD1_Force : BOOL;
- Digitax_ST_Target_position_QD1_Value : DINT;
- Digitax_ST_Status_word_IW0_Force : BOOL;
- Digitax_ST_Status_word_IW0_Value : UINT;
- Digitax_ST_Position_actual_value_ID1_Force : BOOL;
- Digitax_ST_Position_actual_value ID1_Value : DINT;

Einschränkungen

- Nur Kanäle, die im "I/O Abbild" Dialog auf eine Variable abgebildet sind (d.h. eine Variable muss in der Spalte "Variable" eingetragen sein, ob es eine neue oder eine bereits existierende ist, spielt keine Rolle), können über die oben beschriebenen impliziten Variablen geforced werden.
- Nicht verwendete Eingänge / Ausgänge sowie solche, die über eine "AT-Deklaration" in einem Applikationsprogramm gemapped sind, können nicht geforced werden.
- Die betroffenen I/O-Kanäle müssen in mindestens einer Task verwendet werden.
- Geforcte I/Os werden im Monitoring nicht gekennzeichnet (Überwachen-Fenster, I/O Abbild Dialog etc.). Der Wert wird nur implizit im I/O-Treiber für das Schreiben auf das Gerät verwendet.
- Geforcte Eingänge werden korrekt mit dem roten Force-Symbol angezeigt, nicht jedoch geforcte Ein/Ausgänge.

Benutzer und Gruppen

Für Geräte, die eine "**Online Benutzerverwaltung**" unterstützen, wird dieser Unterdialog des Geräte-Editors angezeigt. Hier können Benutzerkonten und Benutzergruppen verwaltet werden, die in Kombination mit der Rechteverwaltung dazu dienen, den Zugriff zur Laufzeit auf Objekte auf der Steuerung zu kontrollieren.

Allgemein

Aus Sicherheitsgründen kann es erwünscht sein, dass bestimmte Funktionen auf der Steuerung nur von autorisierten Benutzern durchgeführt werden können. Zu diesem Zweck bietet die "Online Benutzerverwaltung" Möglichkeiten, Benutzerkonten aufzusetzen, Zugriffsrechte für Benutzergruppen zu vergeben und eine Anmeldung des Benutzers beim Zugriff auf die Steuerung zu erzwingen.

Die gerätespezifische Benutzerverwaltung kann durch die Gerätebeschreibung vordefiniert sein und es hängt auch von dieser Beschreibung ab, inwieweit die vorgegebenen Definitionen in den Konfigurationsdialogen im Programmiersystem bearbeitet werden können.

Wie in der Projekt-Benutzerverwaltung müssen Benutzer Mitglieder von Benutzergruppen sein und können nur Benutzergruppen bestimmte Zugriffsrechte erhalten.

Verwenden des Konfigurationsdialogs

Grundsätzlich ist die Handhabung des Dialogs 'Benutzer und Gruppen' dem der Projekt-Benutzerverwaltung sehr ähnlich. Es gibt sogar die Möglichkeit, Benuterkonto-Definitionen aus der Projekt-Benutzerverwaltung zu "importieren".

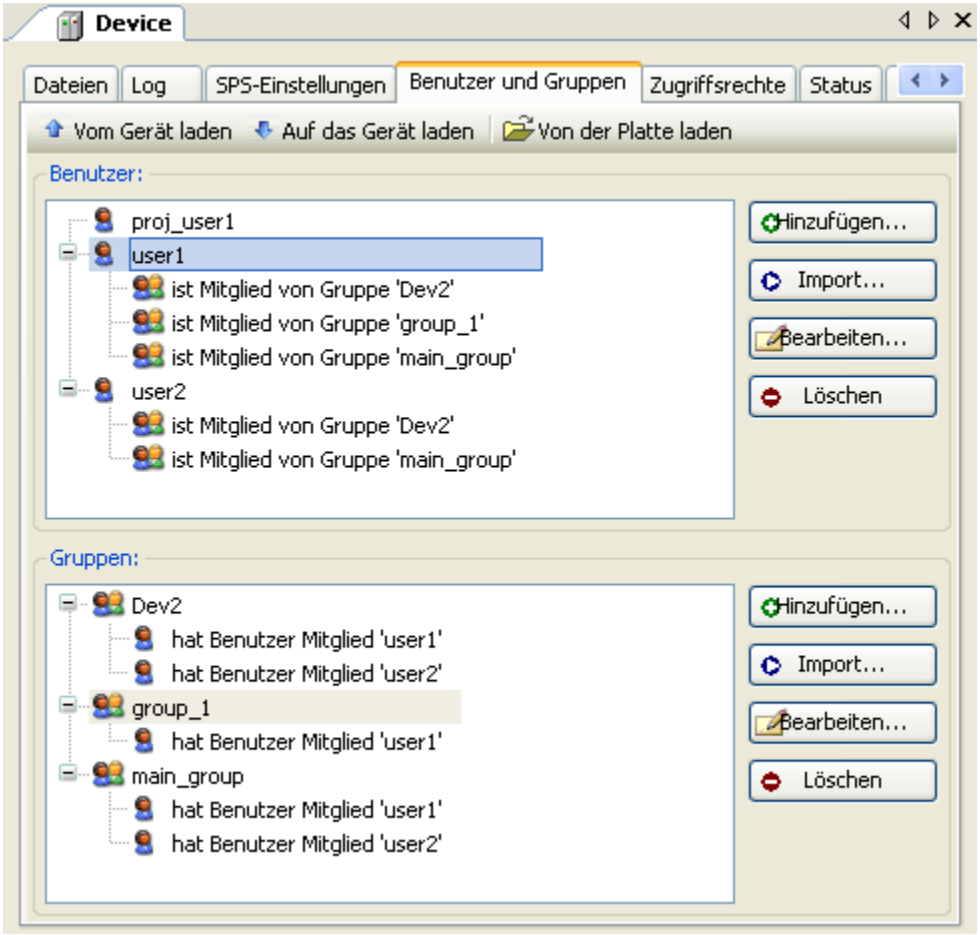


Abb. 467 Geräte-Dialog, Benutzer und Gruppen

Benutzer:

Die folgenden Schaltflächen sind für das Anlegen von Benutzerkonten verfügbar:

Hinzufügen... Der Dialog '**Benutzer hinzufügen**' öffnet, in dem man einen neuen **Benutzernamen** und ein **Passwort** vergeben kann. Das Passwort muss in der **Passwortbestätigung** exakt wiederholt werden.

Abb. 468 Dialog 'Benutzer hinzufügen'

Import... Der Dialog '**Benutzer importieren**' öffnet, in dem alle gerade in der Projekt-Benutzerverwaltung definierten Benutzer aufgelistet sind. Wählen Sie den oder die gewünschten Einträge und bestätigen mit OK um sie in die gerätespezifische Benutzerverwaltung zu übernehmen. Beachten Sie jedoch, dass aus Sicherheitsgründen die **Passwörter nicht importiert** werden!

Hinweis: Jedes importierte Benutzerkonto enthält grundsätzlich eine leere Passwort-Definition!

Bearbeiten... Die Daten des gerade selektierten Benutzers können hinsichtlich Benutzername und Passwort modifiziert werden. Dazu öffnet der Dialog '**Benutzer <Benutzername> bearbeiten**', der dem Dialog 'Benutzer hinzufügen' (s.o.) entspricht.

Löschen Das Konto des gerade ausgewählten Benutzers wird gelöscht.

Gruppen:

Hinzufügen... Der Dialog '**Gruppe hinzufügen**' öffnet. Hier kann ein Name für eine neue Gruppe angegeben werden und aus der Liste der aktuell definierten Benutzer diejenigen ausgewählt werden, die der Gruppe angehören sollen.

Abb. 469 Dialog 'Gruppe hinzufügen'

Import... Der Dialog '**Gruppen importieren**' öffnet, wo die aktuell in der Projekt-Benutzerverwaltung angelegten Gruppen aufgelistet werden. Wählen Sie die gewünschte(n) aus und bestätigen mit OK, um sie in die Liste der gerätespezifischen Benutzerverwaltung zu übernehmen.

Bearbeiten... Die gerade selektierte Gruppe kann hinsichtlich Gruppenname und Mitgliedern bearbeitet werden. Dazu öffnet der Dialog '**Gruppe <Gruppename> bearbeiten**', der dem Dialog 'Gruppe hinzufügen' entspricht..

Löschen Die gerade ausgewählte Gruppe wird gelöscht.

Anwenden und Speichern der aktuellen Konfiguration:

Sehen Sie die entsprechenden Schaltflächen in der Leiste oben im Dialog:

Auf das Gerät laden, Vom Gerät laden: Um die aktuelle Benutzerverwaltungs-Konfiguration wirksam zu machen, muss sie auf das Steuerungsgerät geladen werden. Umgekehrt kann die aktuell auf dem Gerät vorliegende Konfiguration auch wieder in den Editor geladen werden.

Auf Platte speichern, Von der Platte laden: Die aktuelle Benutzerverwaltungs-Konfiguration kann in einer xml-Datei (***.dum**) gespeichert und aus dieser Datei wieder in den Editor geladen werden. Dies kann nützlich sein, wenn die gleiche Benutzer- und Gruppenkonfiguration auf mehreren Systemen eingerichtet werden soll. Der Standard-Dialog zum Navigieren im Dateisystem wird für diesen Zweck verwendet. Der Dateifilter ist automatisch auf "***.dum**" gesetzt, was "Device User Management"-Dateien bedeutet.

Die aktuellen Einstellungen können zur Dokumentation ausgedruckt werden: entweder durch Aufruf des Kommandos "Drucken..." (Datei Menü) oder aber des Kommandos "Dokumentieren..." (Projekt Menü).

Zugriffsrechte

Dieser Unterdialog des Geräte-Editors ist Teil der Funktionalität "Online-Benutzerverwaltung". Hier können den aktuell definierten Benutzergruppen bestimmte Rechte zugewiesen bzw. versagt werden und damit die Zugriffsrechte der Benutzer bzgl. Dateien oder Objekte (z.B. Applikation) auf dem Steuerungsgerät zur Laufzeit festgelegt werden.

Beachten Sie, dass diese Rechte **nur Gruppen zugewiesen** werden können, nicht einzelnen Benutzern. Deshalb muss jeder Benutzer als Mitglied einer Gruppe definiert sein. Die Konfiguration von Benutzern und Gruppen erfolgt im Unterdialog Benutzer und Gruppen des Geräte-Editors.

Sehen Sie ein Beispiel in der folgenden Abbildung: Der Benutzergruppe für Gruppe "main_control" wird das Recht gegeben, dem PlcLogic Knoten "Kindobjekte hinzuzufügen und zu entfernen".

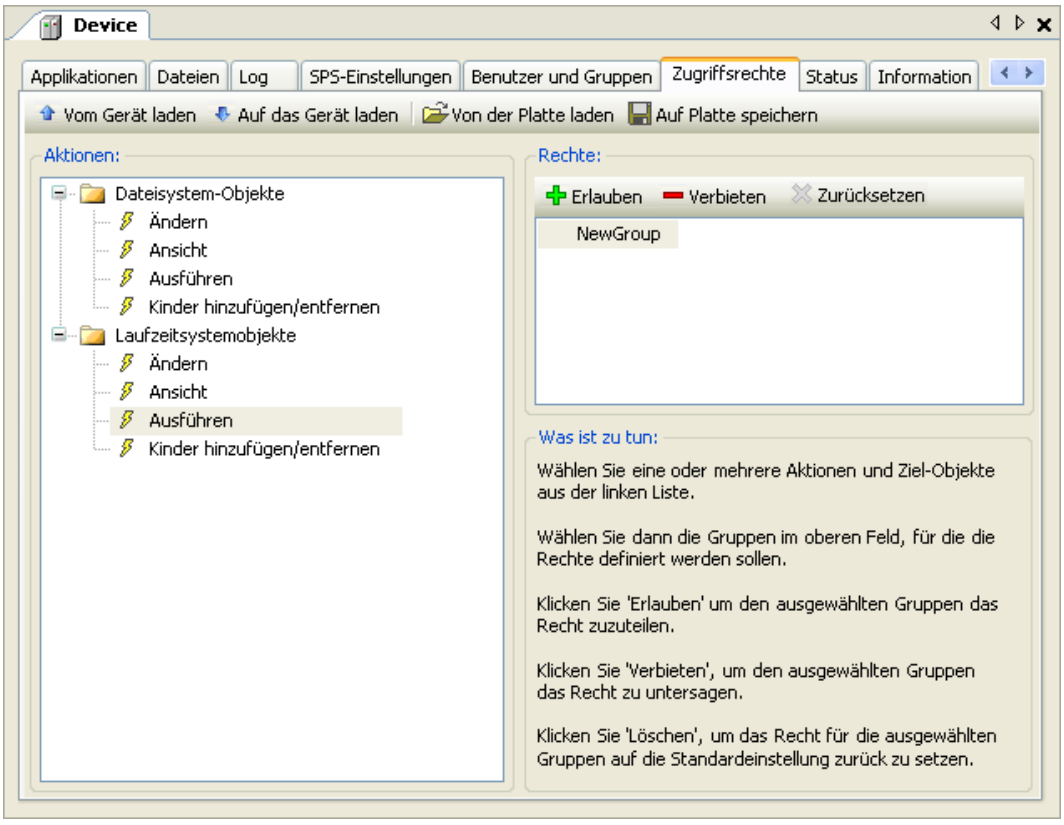


Abb. 470 Geräte-Dialog, Zugriffsrechte

Sehen Sie im folgenden, wie Rechtezuweisungen festgelegt werden und wie sie auf das Gerät geladen oder in einer Datei gespeichert werden können.

Festlegung der Rechte

Um das Recht zu vergeben eine Aktion auf ein oder mehrere Objekte auszuführen, wählen Sie zunächst die Objekteinträge unterhalb der betreffenden Aktion im Aktionen-Fenster aus, selektieren dann die gewünschte Gruppe im 'Rechte' Fenster und drücken dann die Schaltfläche Erlauben oder Verbieten (ebenso im Fenster 'Rechte').

Sehen Sie dazu die Anweisungen im Fenster Was ist zu tun.

Sehen Sie im folgenden die Beschreibungen der einzelnen Fenster:

Aktionen:

Hier werden die möglichen Aktionen aufgelistet, die zur Laufzeit auf Dateien bzw. Laufzeitsystemobjekte (wie z.B. Applikationen) ausgeführt werden können. Der Baum ist folgendermaßen strukturiert:

Objektkategorien:

Auf oberster Ebene gibt es zu rein strukturellen Zwecken zwei Ordner: **Dateisystem-Objekte** und **Laufzeitsystemobjekte**.

Aktionstypen:

Jeweils unter einem Ordner hängen Knoten für die vier **Typen von Aktionen**, die auf ein Objekt ausgeführt werden könnten. Diese Knoten dienen ebenfalls nur strukturellen Zwecken:

Ändern (z.B. Laden der Applikation)

Ansicht (Monitoring)

Kinder hinzufügen/entfernen (Kindobjekte einem existierenden Objekt hinzufügen oder von diesem entfernen)

Ausführen (z.B. Start/Stop der Applikation, Setzen von Haltepunkten)

Objekte (Ziele der Aktion):

Unterhalb jedes Aktionstyp-Knotens finden sich schließlich die "Ziele" der Aktion, d.h. die Laufzeitsystemobjekte oder -Dateien, z.B. Device.
Diese Einträge werden in der einer Baumstruktur dargestellt, die dem Gerätebaum bzw. der Dateisystemstruktur entspricht.

Hinweis: Beachten Sie, dass das Zuweisen einer Rechtedefinition zu einem "Vaterobjekt" im Objektbaum üblicherweise bedeutet, dass die "Kindobjekte" die Definition erben, solange sie nicht explizit eine eigene Zuweisung erhalten. Allerdings kann dies auf geräteabhängig auch anders gehandhabt werden; deshalb informieren Sie sich besser diesbezüglich vor der Rechtevergabe. Vererbungen werden hier im Dialog nicht dargestellt.

Rechte:

Hier werden die aktuell definierten Benutzergruppen angezeigt. Vor jeder Gruppe zeigt eines der folgenden Symbole die aktuelle Rechtedefinition bezüglich des gerade im Aktionen-Fenster ausgewählten Zielobjekts an:

: Die gerade im Aktionen-Fenster ausgewählte(n) Aktion(en) sind dieser Gruppe erlaubt.

: Die gerade im Aktionen-Fenster ausgewählte(n) Aktion(en) sind für diese Gruppe verboten.

: Im Moment gibt es keine explizite Rechtedefinition für die gerade im Aktionen-Fenster ausgewählte(n) Aktion(en).

Wenn gerade mehrere Aktionen im Aktionen-Fenster ausgewählt sind, die keine einheitliche Definition bezüglich der gerade ausgewählten Gruppe haben, wird kein Symbol angezeigt.

Schaltflächenleiste: Nachdem Sie die gewünschten Objekt(e) unterhalb der gewünschten Aktion im Aktionen-Fenster und die gewünschte Gruppe im Rechte-Fenster ausgewählt haben, klicken Sie auf eine der Schaltflächen:

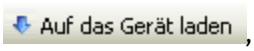
Erlauben: Der Zugriff (Aktion) wird explizit erlaubt.

Verbieten: Der Zugriff (Aktion) wird explizit verboten.

Zurücksetzen: Der aktuell erlaubte Zugriff für die gerade im Aktionen-Fenster ausgewählten Aktion(en) wird zurückgenommen, d.h. auf den Standardwert zurückgesetzt.

Anwenden und Speichern der aktuellen Konfiguration:

Sehen Sie die entsprechenden Schaltflächen in der Leiste oben im Dialog:



,

: Um die aktuelle Konfiguration der Zugriffsrechte wirksam zu machen, muss sie auf das Steuerungsgerät geladen werden. Umgekehrt kann die aktuell auf dem Gerät vorliegende Konfiguration auch wieder in den Editor geladen werden.

,

: Die aktuelle Rechtekonfiguration kann in einer xml-Datei (***.dar**) gespeichert und aus dieser Datei wieder in den Editor geladen werden. Dies kann nützlich sein, wenn die gleiche Konfiguration auf mehreren Systemen eingerichtet werden soll. Der Standard-Dialog zum Navigieren im Dateisystem wird für diesen Zweck verwendet. Der Dateifilter ist automatisch auf "***.dar**" gesetzt, was "Device Access Rights"-Dateien bedeutet.

Die aktuellen Einstellungen können zur Dokumentation ausgedruckt werden: entweder durch Aufruf des Kommandos "Drucken..." (Datei Menü) oder aber des Kommandos "Dokumentieren..." (Projekt Menü).

Status

Dieser Unterdialog des Geräte-Dialogs (Geräte-Editor) zeigt Status-Informationen (z.B. "Running", "Stopped") und spezifische Diagnosemeldungen aus dem jeweiligen Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Information

Dieser Unterdialog des **Geräte-Dialogs** (Geräte-Editor) zeigt für das gerade im Gerätebaum ausgewählte Gerät einige allgemeine Informationen an, die aus der Gerätebeschreibungsdatei stammen: Name, Hersteller, Kategorien, Version, Bestellnummer, Beschreibung, ggfs. Abbildung.

14.2 CANopen-Konfigurationseditor

14.2.1 CANbus

Das **CANbus** Objekt muss als oberster Knoten einer CANbus Konfiguration in den Gerätebaum eingetragen werden.

Im zugehörigen Konfigurationsdialog wird folgendes definiert:

Baudrate (bits/s): Baudrate in Bits pro Sekunden für die Datenübertragung auf dem Bus. Der Standardwert wird aus der Gerätebeschreibungsdatei (*.devdesc) des CANbus Geräts entnommen. Die folgenden Baudraten können eingestellt werden: 10000, 20000, 50000, 100000, 125000, 250000, 500000, 800000 und 1000000. Die Einheit ist bits/s (Bits pro Sekunde) wobei in den Konfigurationsdaten die Baudrate in KBit/s gespeichert wird.

Netzwerk: Nummer des CAN-Netzwerks, das über die CANbus-Schnittstelle angebunden werden soll. Erlaubte Werte: 0 bis 100.

Online Bus Access

Bei laufender Applikation 0SDO und NMT Zugriff blockieren: Wenn diese Option aktiviert ist, ist es dem Anwender nicht möglich mit dem CANopen Slave zu kommunizieren während die IEC Applikation auf dem Zielgerät läuft (z.B. Onlinedienste auf der Statusseite)

CANopen Konfigurationseditor

Das Plug-In 'DeviceEditorCANbus' stellt Dialoge für die Konfiguration von CANopen Geräten und Modulen gemäß der Norm CANopen Draft Standard 301 bereit.

Zum Abbilden einer CANbus-Hardware im Projekt werden die entsprechenden Geräteobjekte im Gerätebaum eingefügt und konfiguriert. Dazu stehen dann die speziellen CANbus Konfigurationsdialoge neben den Standarddialogen des Geräte-Editors zur Verfügung.

Die CANbus Geräte und Module erhalten eine Konfiguration, die ihr Zeit- und fehlerbezogenes Verhalten während des Datenaustauschs definiert. Weiterhin wird für jedes Modul das Abbilden ("Mapping") ihrer Eingänge und Ausgänge auf **PDO**s (Process Data Objects) vorgenommen, die für das Senden und Empfangen der Prozessdaten verwendet werden.

Danach wird eine Liste von **SDO**s (Service Data Objects) erzeugt, die alle Konfigurationsdaten - mit Hilfe von Anforderungs- und Antwortdiensten - auf das CANbus Zielsystem übertragen. Dort werden die Kommunikations- und Prozessdaten im "Objektverzeichnis" verwaltet.

Die CANopen Modulkonfiguration basiert auf den Gerätebeschreibungsdateien der verwendeten Master- und Slave-Geräte und kann vom Anwender in den Konfigurationsdialogen noch angepaßt werden. Beachten Sie die Empfehlung, für Standardapplikationen und modulare Geräte den Autokonfigurationsmodus zu verwenden!

Folgende Schritte müssen also durchgeführt werden, um eine CANbus-Konfiguration im Gerätebaum des Projekts anzulegen:

1. Installieren der erforderlichen Gerätebeschreibungsdateien
2. Sofern der CANbus in Verbindung mit dem SJA Chip verwendet wird, muss die »PLC Designer« Konfigurationsdatei (...\GatewayPLC\»PLC Designer«SP.cfg) um folgenden Eintrag ergänzt werden:
   Component.<subsequent number>=CmpSJACanDrv
3. Abbilden der CANbus-Struktur durch Einfügen der entsprechenden Geräteobjekte im Geräte-Fenster.
4. Überprüfen bzw. Anpassen der Konfigurationsparameter in den Dialogen des Geräte-Editors.

Installieren der CANopen Gerätedateien:

Um die CANbus Hardware im Gerätebaum des Projekts abbilden zu können, müssen die entsprechenden **Gerätebeschreibungsdateien** über die Dialoge des Geräte-Repositories auf dem lokalen System installiert werden. Dies betrifft die Beschreibungsdateien (***.devdesc.xml**) für die Basismodule "CANbus" und "CANopen_Manager", die mit dem Standard-CANbus-Konfigurations-Plug-In bereitgestellt werden, wie auch die herstellerspezifischen **EDS-Dateien** (Electronic Data Sheet) für die CANopen Geräte und Module, die im Bussystem verwendet werden. Diese Dateien enthalten Parameterdefinitionen und Konfigurationsoptionen eines Moduls.

Basis-Hierarchie einer CANbus Konfiguration in **»PLC Designer« / Einfügen der** CANbus Geräteobjekte :

- Der Basisknoten, d.h. der oberste Eintrag in einem CANbus Konfigurationsbaum, muss das "**CANbus**" Objekt sein. Ein CANbus Objekt kann nur unterhalb eines programmierbaren Geräts eingehängt werden. Also selektieren Sie ein solches im Gerätebaum und wählen "CANbus" im Dialog 'Gerät anhängen' oder 'Objekt hinzufügen':

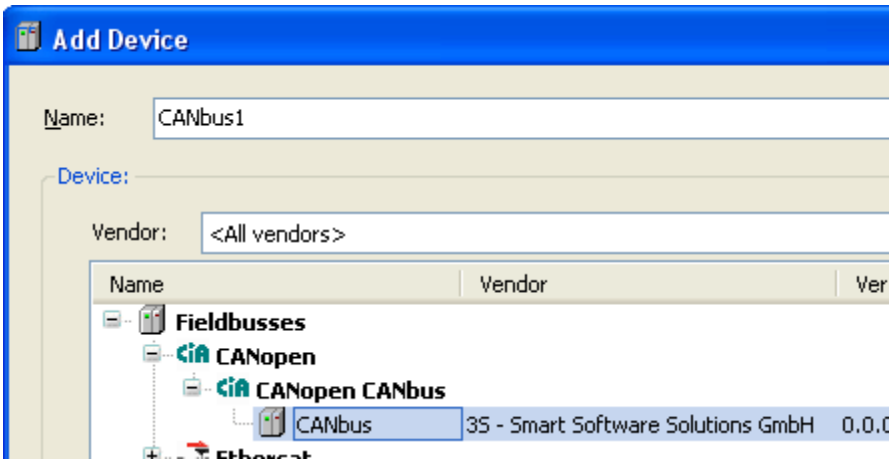


Abb. 471 CANbus Objekt einfügen

- Im CANbus Knoten müssen die 'Baudrate' für den Datentransfer auf dem Bus und die Nummer des CAN-Netzwerks, das über die CANbus-Schnittstelle angebunden werden soll überprüft bzw. angepasst werden.
- Unterhalb eines CANbus Eintrags muss das obligatorische Objekt "**CANopen Manager**" eingefügt werden. Dieser Manager kann als "Master" des CANbus gesehen werden. Er unterstützt die CANbus-Konfiguration derzeit ausschließlich durch implizite Funktionen und somit ist keine weitere Konfiguration durch den Anwender erforderlich. Der CANopen Manager erleichtert das Aufsetzen der Verbindung mit dem Netzwerk, speziell durch das Vereinfachen der PDO-Verknüpfungen zwischen den einzelnen Modulen.
- Unterhalb des CANopen Managers werden schließlich die einzelnen **CANopen Gerät**eobjekte ("Slaves") eingefügt, die mit der Hardware bereitgestellt werden. Hier sind zwei Typen möglich: nicht-modulare Geräte und modulare Geräte. Unterhalb von modularen Geräten können weitere **CAN-Module** eingehängt werden und automatisches I/O-Mapping kann verwendet werden.

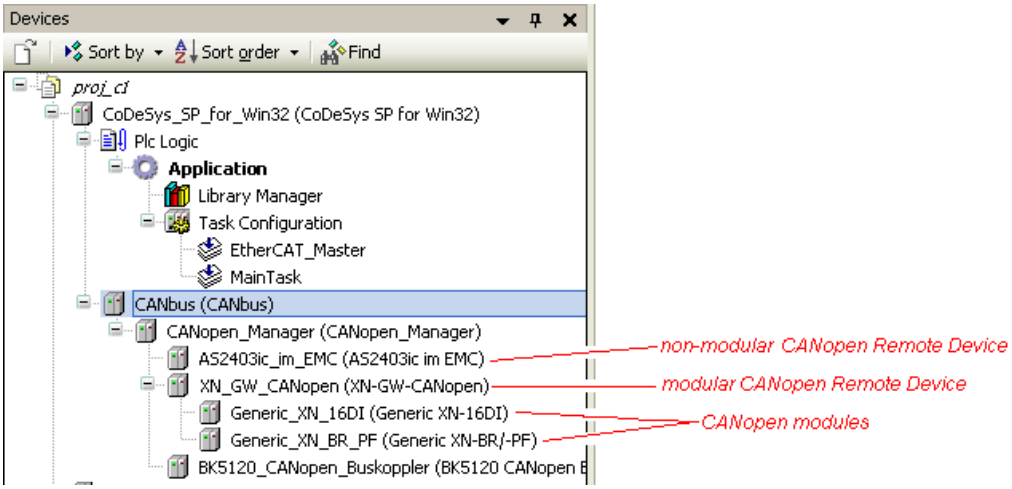


Abb. 472 Beispiel: CANbus Konfigurationsbaum

Konfiguration

Nachdem die CANbus Hardware im Gerätebaum abgebildet ist, verwenden Sie die Konfigurationsdialoge für die einzelnen Objekte, um die Standardeinstellungen zu prüfen bzw. ggfs. anzupassen:

Für das gerade im Baum selektierte Objekt erscheint der passende Konfigurationsdialog im Fenster des Geräte-Editors, wenn sie einen Doppelklick auf den Eintrag ausführen oder den Befehl 'Objekt bearbeiten' wählen.

Die Konfigurationsmöglichkeiten eines Geräts oder Moduls hängen immer von den Definitionen in der Gerätebeschreibungsdatei ab.

Beachten Sie die allgemeinen Einstellungen für den Geräte-Editor im Optionen-Dialog: Beispielsweise können die generischen Dialoge ausgeblendet werden.

Ein CANopen Konfigurationsdialog ist immer mit dem Namen des jeweiligen Moduls/Geräts betitelt (wie im Gerätebaum verwendet), und kann die folgenden Unterdialoge enthalten:

Für das CANbus Objekt:

- CANbus (allgemeine Einstellungen, derzeit nur Baudrate)
- CANbus Konfiguration (Einstellungen wie von der *.devdesc.xml-Datei vorgegeben)
- Information

Für das CANopen Manager Objekt:

- CANopen Manager
- Common.CANbus Konfiguration
- CANopen I/O Abbild
- Status
- Information

Für ein CANopen Gerät:

- CANopen Remote Device (Haupteinstellungen des Geräts)
- PDO Mapping
- Receive PDO Mapping
- Send PDO Mapping
- Service Data Object
- CANopen Konfiguration (Parameter, wie in der EDS-Datei definiert)
- CANopen I/O Mapping
- Information

Für ein CANopen Modul:

- CANopen-Module Konfiguration (Parameter, wie in der EDS-Datei definiert)
- CANopen-Module I/O-Mapping
- Information

Schnittstellen für ein CANopen Gerät

Um ein CANopen Gerät aus der IEC Applikation heraus anzusprechen, kann die von der Bibliothek IIoDrvBusControl.library zur Verfügung gestellte Schnittstelle implementiert werden.

CANbus Knoten

CANbus

Das **CANbus** Objekt muss als oberster Knoten einer CANbus Konfiguration in den Gerätebaum eingetragen werden.

Im zugehörigen Konfigurationsdialog wird folgendes definiert:

Baudrate (bits/s): Baudrate in Bits pro Sekunden für die Datenübertragung auf dem Bus. Der Standardwert wird aus der Gerätebeschreibungsdatei (*.devdesc) des CANbus Geräts entnommen. Die folgenden Baudraten können eingestellt werden: 10000, 20000, 50000, 100000, 125000, 250000, 500000, 800000 und 1000000. Die Einheit ist bits/s (Bits pro Sekunde) wobei in den Konfigurationsdaten die Baudrate in KBit/s gespeichert wird.

Netz: Nummer des CAN-Netzwerks, das über die CANbus-Schnittstelle angebunden werden soll. Erlaubte Werte: 0 bis 100.

CANbus Konfiguration

Dieser Konfigurationsdialog für einen CANbus Knoten ist derzeit ohne Funktion, da die Gerätebeschreibung außer der Baudrate und Netzwerknummer keine weiteren Konfigurationsparameter enthält.

Status

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Status-Informationen (z.B. "Running", "Stopped") und spezifische Diagnosemeldungen aus dem Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Information

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Informationen zum Gerät bzw. Modul, die aus der Gerätebeschreibung gelesen werden. Mögliche Angaben: Name, Hersteller, Typ, Versionsnummer, Kategorien, Bestellnummer, Abbildung.

CANopen Manager

Der **CANopen Manager** ist in einer CANbus Konfiguration ein obligatorischer Knoten unterhalb des CANbus Knotens. Dieser "Manager" unterstützt die CANbus Konfiguration durch interne Funktionen und dient daher gewöhnlich als CANbus Master.

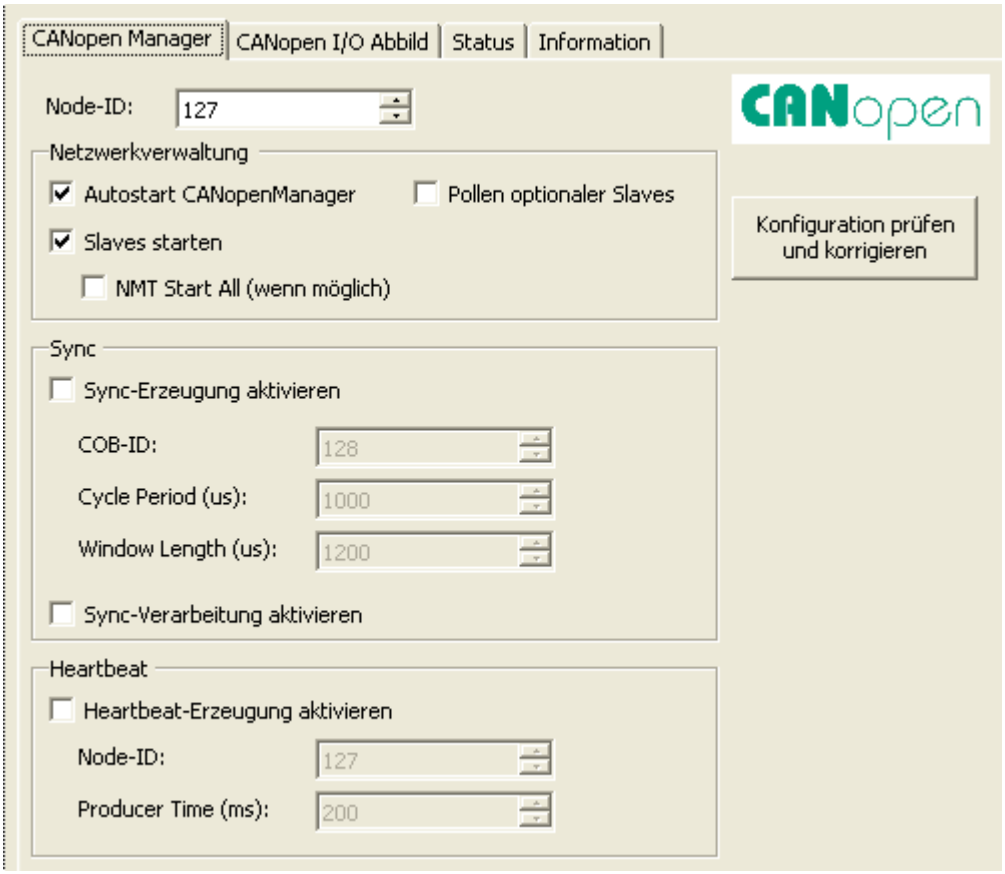


Abb. 473 CANopen Manager Dialog

Im CANopen Manager Dialog kann die synchronisierte Übertragung für die PDO Übermittlung auf den Bus eingeschaltet werden. Dort sind auch einige Standardwerte definiert, welche auf den speziellen Gebrauchsfall angepasst werden können innerhalb der Common.CANbus Konfiguration:

Netzwerkverwaltung:

Autostart CANopen Manager: Diese Option legt fest, ob der CANopen Manager automatisch in den Status ‚Operational' schalten soll. Default: Aktiv.

Slaves starten: Definiert, ob der CANopen Manager die Slaves starten soll. Default: Aktiv.

NMT Start All: Ist diese Option gesetzt, so wird das Start Node Kommando als Broadcast Kommando abgesetzt. Ist die Option nicht gesetzt, so werden einzelne Start Node Kommandos an jeden Slave gesendet. Default: Nicht aktiv.

Bitte beachten Sie: Einige Slaves unterstützen das Broadcast Start Node Kommando nicht, z.B. der Lenze ECS Antrieb.

Pollen optionaler Slaves: Optionale Slaves werden zyklisch per SDO Leseauftrag angefragt. Diese Funktion wird nur dann benötigt, wenn der optionale Slave von sich aus kein Boot-Telegramm nach dem Einschalten sendet. Default: Nicht aktiv.

Sync:

Sync-Erzeugung aktivieren: Diese Option ist standardmäßig deaktiviert. Nach ihrer Aktivierung wird die Übertragung auf den Bus synchronisiert unter Berücksichtigung folgender Einstellungen:

Sync-Zykluszeit (µs): Intervalllänge (in Mikrosekunden), nach welcher die Synchronisationsnachricht verschickt wird. Möglicher Wertebereich: [100, 2^{32}-1].

Sync COB-ID: Identifizierung des Kommunikationsobjekts als Synchronisationsnachricht. Möglicher Wertebereich: [1, 2047].

Überwachungsfenster (µs): enthält die Länge des Zeitfensters für synchrone PDOs in Mikrosekunden. Mögliche Werte: [1,2^{32}-1] bzw. 0, falls ungenutzt.

Die synchronen PDO's werden direkt nach der Synchronisationsnachricht verschickt.

Sync-Verarbeitung aktivieren: Diese Option ist standardmäßig deaktiviert. Ihre Aktivierung zeigt an, dass ein anderes Gerät Synchronisationsnachrichten erzeugt, welche vom CANopen Manager empfangen werden.

Hinweis: Bei Aktivierung der Sync-Erzeugung für den CANopen Manager darf bei allen anderen Busteilnehmern die Option "Zyklischer Betrieb" nicht mehr angewendet werden.

Heartbeat:

Das Arbeiten mit Heartbeats ist ein alternativer Mechanismus zur Überwachung: Im Gegensatz zu der Nodeguarding Funktionalität kann er sowohl von Mater- als auch von Slaveknoten ausgeführt werden. Für gewöhnlich sendet der Master Heartbeats zu den Slaves.

Heartbeat-Erzeugung aktivieren: Bei Aktivierung dieser Option sendet der Master Heartbeats gemäß des in der Heartbeat Time definierten Zeitintervalls. Werden nun neue Slaves mit Heartbeat Funktionalität angehängt, so wird ihr Heartbeat Verhalten automatisch aktiviert und entsprechend konfiguriert, d.h. die Node-ID des Master wird übernommen und sein Heartbeat-Intervall mit dem Faktor 1,2 multipliziert. Ist die Heartbeat-Erzeugung des CANopen Manager nicht aktiviert, so wird beim Slave stattdessen Node-Guarding aktiviert mit dem Life Time Faktor 10 und einer Guard Time von 100ms.Beachten Sie, dass auch ein CANopen (slave) Gerät als Heartbeat-Erzeuger konfiguriert werden kann.

Node-ID: Eindeutige Kennzeichnung (1-27) des Heartbeat-Erzeugers auf dem Bus.

Heartbeat-Intervall (ms): Intervalllänge zwischen aufeinanderfolgenden Heartbeats in Millisekunden.

Werden mehrere Geräte unterhalb des CANopen Manager eingehängt, so können Fehlermeldungen erscheinen, welche mehrfach vergebene Node IDs oder unzulässige COB-IDs melden. Die Schaltfläche **Konfiguration prüfen und korrigieren** erlaubt eine Auflösung dieser Konflikte im öffnenden Dialog:

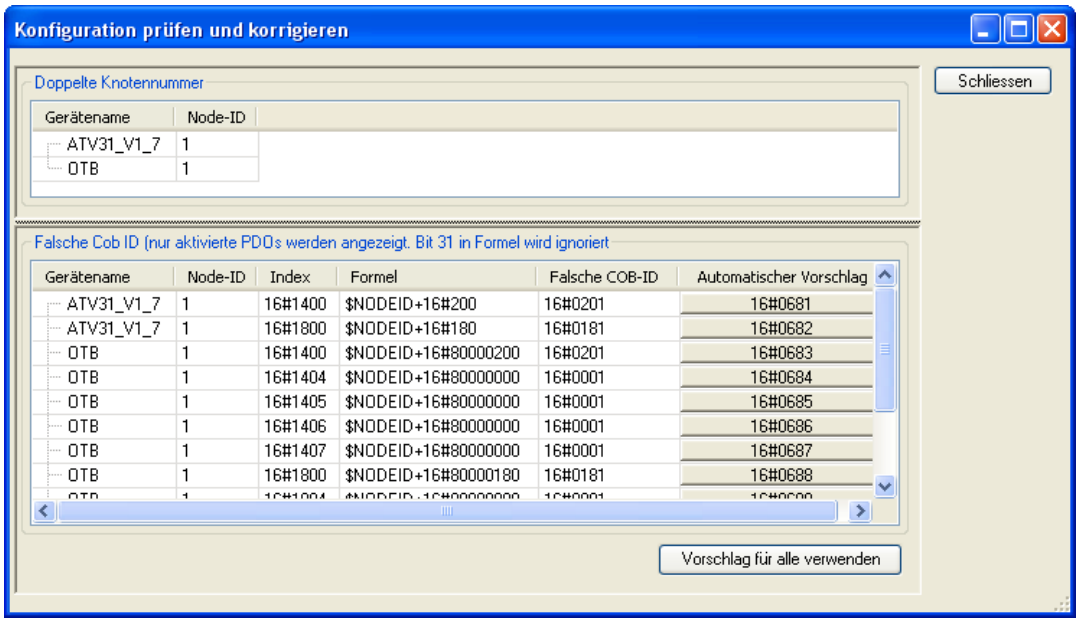


Abb. 474 Dialog - Konfiguration prüfen und korrigieren

Doppelte Knotennummer: Dieses Feld enthält eine Liste aller Geräte, die identische IDs tragen. In den Feldern der Spalte "Node-ID" können Sie neue Knotennummern für die zugehörigen Geräte eintragen.

Falsche COB-ID (nur aktivierte PDOs werden angezeigt). Bit 31 in Formel wird ignoriert: Die aus den Gerätebeschreibungsdateien automatisch erzeugten COB-IDs können unzulässig sein. Alle inkorrekten Einträge werden mit den zugehörigen Gerätenamen, Node-IDs und Indices aufgelistet. Drei Möglichkeiten zur Korrektur der ungültigen COB-IDs stehen zur Verfügung:

- Entweder kann die angezeigte Formel zur Berechnung der COB-IDs korrigiert werden, so dass eine gültige COB-ID resultiert. Das Abändern der Formel kann direkt innerhalb des entsprechenden Tabellenfelds vorgenommen werden.
- Oder Sie übernehmen den "Automatischen Vorschlag" für die COB-ID, indem Sie auf den entsprechenden Eintrag in der letzten Tabellenspalte klicken.
- Oder aber Sie akzeptieren alle automatisch erzeugten Vorschläge aus der letzten Tabellenspalte auf einmal durch Betätigen der Schaltfläche **Vorschlag für alle verwenden.**

Korrigierte Einträge werden automatisch aus den angezeigten Listen entfernt.

Common.CANbus Konfiguration

Der zugehörige Dialog ist im Geräte Editor des CANopen Managers nur dann verfügbar, wenn die Option 'Generische Konfigurationseditoren anzeigen' aktiviert wurde. Der Dialog korrespondiert zum Dialog Konfiguration, der vom Geräte Editor auch für andere Gerätemodule bereitgestellt wird.

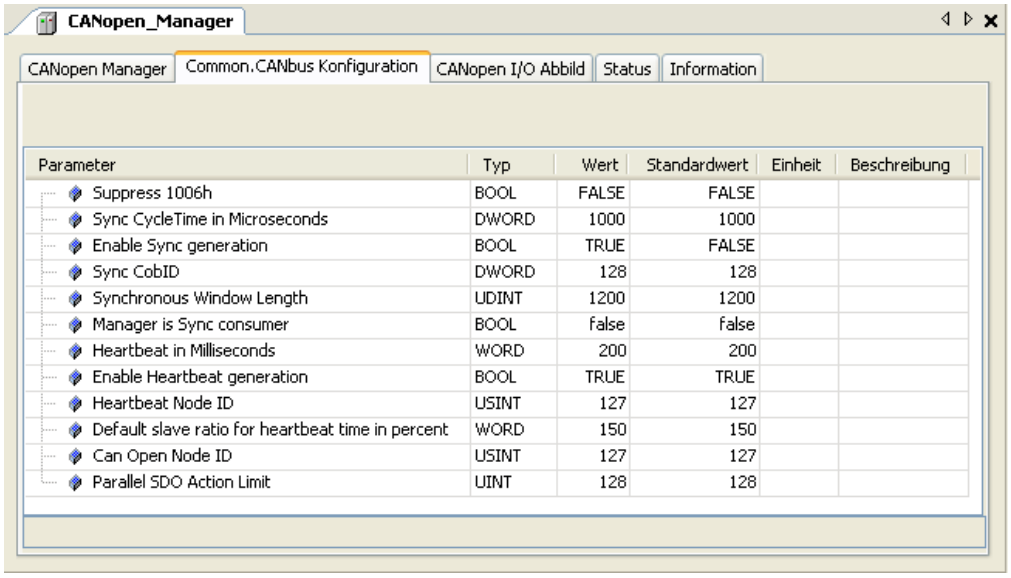


Abb. 475 Common.CANbus Konfiguration Dialog

Die vom Hersteller definierten gerätespezifischen Parameter der Gerätebeschreibungsdatei sind in einer Tabelle aufgeführt und können dort editiert werden, sofern dies von der Beschreibungsdatei erlaubt wird. Für die meisten davon können die Standardwerte innerhalb des CANopen Manager Dialogs gesetzt werden, die folgenden sind jedoch nur hier im Common.CANbus Konfigurationsdialog verfügbar:

Suppress 1006h: Wenn TRUE, wird kein Objekt 0x1006 geschrieben, auch wenn die Sync-Erzeugung aktiviert ist. Wenn FALSE oder nicht vorhanden, wird Objekt 0x1006 wie in früheren Versionen geschrieben. Standardwert: FALSE.

Default slave ratio for heartbeat time in percent (Parameter ID 275): Wenn **Heartbeat-Erzeugung aktivieren** für den CANOpen Manager aktiviert ist, wird dieser Wert zur Berechnung der Standard Heartbeat Consumer Zeit eines Slaves verwendet. Standardwert: 150.

Optional Device Default (Parameter ID 276): Standardeinstellung für die Optionenbox “Optionales Gerät” im Konfigurationsdialog eines Slaves. Standardwert: FALSE.

Parallel SDO Action Limit: Anzahl der SDO Aktionen, die gleichzeitig bearbeitet werden können, d.h. die Anzahl der SDO-Kanäle des CANopen Managers. Dies betrifft sowohl die für die Konfiguration verwendeten SDOs als auch die von der Applikation angeforderten. Standardwert: 128.

CANopen I/O Abbild

Bitte sehen Sie Geräteeditor, E/A-Abbild, um eine allgemeine Beschreibung und Anleitung zum "Mapping"-Dialog zu erhalten.

Status

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Status-Informationen (z.B. "Running", "Stopped") und spezifische Diagnosemeldungen aus dem Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Information

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Informationen zum Gerät bzw. Modul, die aus der Gerätebeschreibung gelesen werden. Mögliche Angaben: Name, Hersteller, Typ, Versionsnummer, Kategorien, Bestellnummer, Abbildung.

CANopen Gerät

Nicht-modulare und modulare Geräte

In einer CANbus Konfiguration können zwei verschiedene Typen von CAN-Geräten unterhalb des CANopen Manager Knotens eingefügt werden:

Unterhalb eines **modularen Geräts** können CAN Module (Submodule) eingehängt werden. Für ein modulares Gerät gibt es keinen CANopen I/O Abbild Dialog, weil die einzelnen Submodule ihre eigenen Mapping-Listen haben. Der Dialog CANopen Remote Device eines modularen Geräts bietet die Option '**PDO Mapping automatisch konfigurieren**', die für Standardapplikationen empfohlen wird.

Unterhalb eines **nicht-modularen Geräts** können keine weiteren Module eingehängt werden. Deshalb gibt es im Unterschied zum modularen Gerät hier einen Dialog I/O Abbild für das Konfigurieren des "Mappings" der Ein- und Ausgänge und keine Option, dies automatisch durchführen zu lassen.

CANopen Remote Device

In diesem Dialog einer CANopen Gerätekonfiguration ist die Knotennummer des Gerätes spezifiziert. Nach Aktivieren des Kontrollkästchens "**Experten-PDO-Einstellungen aktivieren**" werden alle Einstellungen, die von der Gerätebeschreibung (EDS-Datei) für das Gerät vorgegeben sind, zunächst mit den Standardwerten angezeigt; sie können vom Anwender verändert werden.

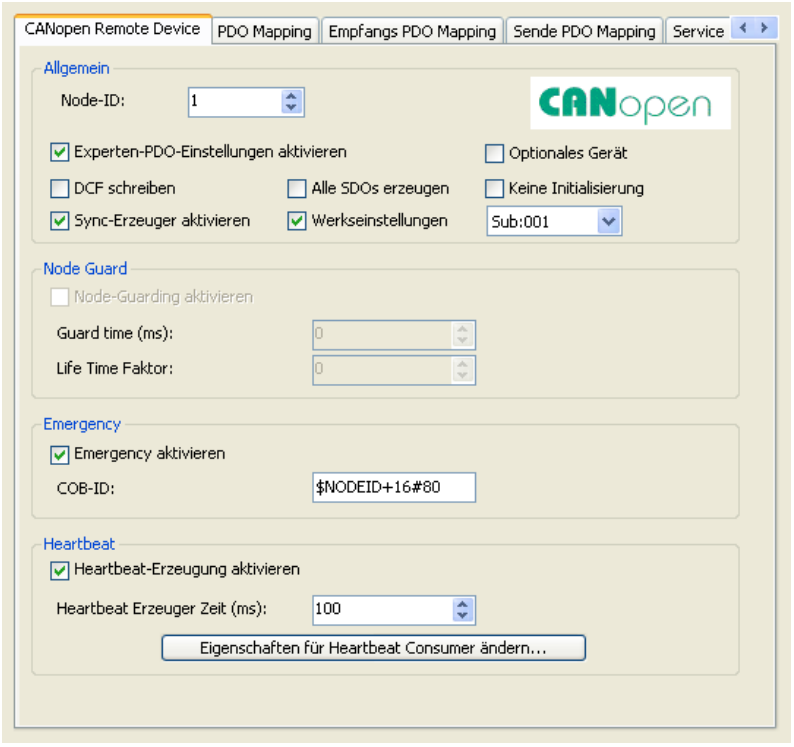


Abb. 476 CANopen Remote Device Konfiguration

Allgemein:

Autokonfig. PDO Mapping: Wenn diese Option aktiviert ist (**was für Standardapplikationen sehr empfohlen wird!**), wird das PDO Mapping (Dialoge Empfangs PDO Mapping und Sende PDO Mapping) automatisch aus den Definitionen in der Gerätebeschreibung erzeugt und kann in den beiden Mapping-Dialogen nicht verändert werden. Diese Option ist für nicht-modulare Geräte nicht verfügbar.

Node-ID: Diese Knotennummer identifiziert das CAN Modul eindeutig und entspricht der auf dem Modul selbst gesetzten Nummer (Wert zwischen 1 und 127). Die Node-ID muss dezimal angegeben sein.

DCF schreiben: Derzeit ohne Funktion!

Alle SDO's erzeugen: Alle Objekte werden auf die Steuerung geladen unabhängig davon ob der Standardwert verändert wurde oder nicht.

Werkseinstellungen: Die Verfügbarkeit dieser Option ist abhängig vom Inhalt der Gerätebeschreibungsdatei. Sofern die Option verfügbar ist, ist sie standardmäßig aktiviert. Bei Aktivierung werden die Slave-Knoten auf die Standardparameter zurückgesetzt, bevor die Konfiguration auf das Gerät geladen wird. Welche Parameter gesetzt werden können, ist gerätespezifisch und die konkrete Angabe erfolgt über die Auswahl eines Subindex in der Auswahlliste neben dem Optionenfeld. Beispielsweise werden standardmäßig mit Subindex 2 (1000h bis 1FFFh, Objekt 1011h) die CANopen Kommunikationseinstellungen angesprochen.

Optionales Gerät: Wenn diese Option aktiviert ist (die Verfügbarkeit hier im Dialog hängt vom Gerät ab), versucht der Master nur einmal von diesem Knoten zu lesen. Dann wird der Knoten ignoriert, wenn er nicht antwortet, d.h. der Master geht in den normalen Betriebszustand über.

Keine Initialisierung: Wenn diese Option aktiviert ist (die Verfügbarkeit hier im Dialog hängt vom Gerät ab), nimmt der Master den Node sofort in Betrieb, ohne ihm Konfigurations-SDOs zu schicken. (Die SDO-Daten werden aber dennoch erzeugt und auf der Steuerung gespeichert.)

Node Guard:

Node-Guarding aktivieren: Ist die Option Nodeguarding aktiviert, so wird im Intervall, das bei **Guard Time** in Millisekunden (standardmäßig 100) angegeben ist, eine Nachricht an das Modul verschickt. Meldet sich das Modul daraufhin nicht mit der angegebenen **Guard COB-ID** (Communication Object Identifier), so erhält es den Status "Timeout". Ist die Anzahl der Versuche (**Life Time Faktor**, standardmäßig auf Wert 10 gesetzt, sofern keine spezielle oder die Angabe 0 in der Gerätebeschreibungsdatei enthalten ist) erreicht, so gilt das Modul als nicht OK. Der Status des Moduls wird bei der Diagnoseadresse hinterlegt. Ist keine Guard Time und kein Life Time Factor angegeben (0), erfolgt keine Überwachung des Moduls.

Emergency:

Emergency aktivieren: Ist diese Option aktiviert, so sendet ein Modul bei internen Fehlern Emergency-Nachrichten mit einer eindeutigen **COB-ID**. Diese je nach Modul unterschiedlichen Nachrichten werden an der Diagnoseadresse hinterlegt.

Heartbeat:

Heartbeat-Erzeugung aktivieren: Wenn diese Option aktiviert ist, sendet das Modul in den bei **Heartbeat Erzeuger Zeit (ms)** (standardmäßig auf Wert 10 gesetzt, sofern keine spezielle oder die Angabe 0 in der Gerätebeschreibungsdatei enthalten ist) angegebenen ms-Abständen Heartbeats aus.

Eigenschaften für Heartbeat Consumer ändern: Diese Schaltfläche öffnet einen Dialog, in dem die in der EDS-Datei definierten Knoten aufgelistet sind. Sie können auswählen, welche davon überwacht werden sollen. Klicken Sie dazu in das jeweilige Feld 'Aktivieren' und geben im Feld 'Heartbeat time' das gewünschte Hearbeat-Intervall ein. Durch einen Doppelklick auf das Feld erhält man dazu ein entsprechendes Eingabefeld. Wenn die Hearbeat Consumer Option aktiviert ist, hört das Modul auf die vom Master gesendeten Heartbeats und schaltet seine Ein- und Ausgänge ab, sobald kein Heartbeat mehr eintrifft.

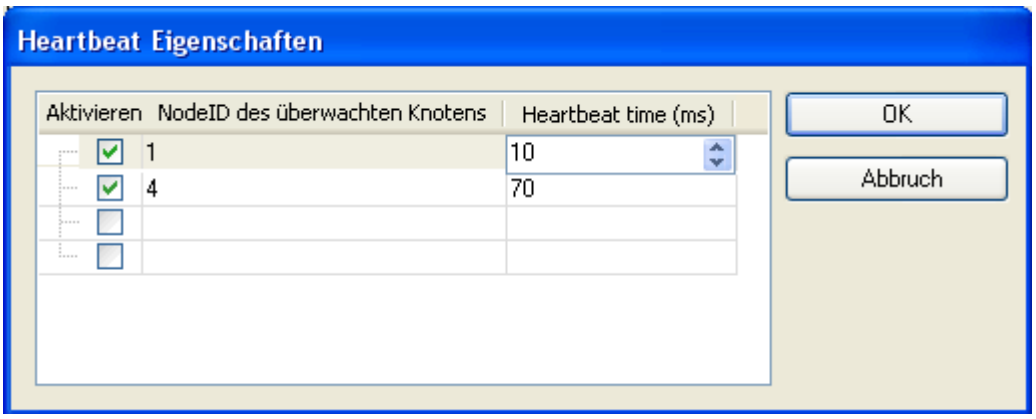


Abb. 477 Heartbeat Eigenschaften Dialog

Hinweis: Wenn ein Gerät mit Heartbeat Funktionalität eingefügt wird, werden seine Heartbeat Einstellungen automatisch auf die des Masters (CANopen Manager) abgestimmt.

PDO Mapping

Dieser Dialog in der CANopen Gerätekonfiguration ist eine alternative und vereinfachte Darstellung der aktuell konfigurierten Sende- und Empfangs-PDOs.

Wenn Schreibrecht für das PDO besteht (Definition in der Gerätebeschreibung, Standard: Schreibrecht), kann das PDO hier im Dialog aktiviert bzw. deaktiviert werden. Bei modularen Geräten muss dazu allerdings der Autokonfigurations-Modus deaktiviert sein!

Der Übertragungsmodus kann hier immer verändert werden. Dafür muss über einen Doppelklick auf den PDO-Eintrag der Dialog "PDO Eigenschaften" geöffnet werden.

Weitere Modifikationen der PDO-Konfiguration können nur in den Dialogen Empfangs PDO Mapping und Sende PDO Mapping vorgenommen werden.

Empfangs PDO Mapping

Dieser Dialog ist nur verfügbar, wenn die Option Enable Expert PDO Settings für das Gerät aktiviert ist.

In diesem Dialog einer CANopen Gerätekonfiguration ist das Mapping für Empfangs-PDOs definiert, d.h. die Zuordnung von **Ausgängen** (Objektverzeichniseinträge) zu einem PDO zum Zweck des Datenaustausches im CAN-Bus. Die entsprechende Zuordnung zu den IEC-Adressen wird im Dialog I/O Abbild sichtbar.

Der Dialog zeigt die einzelnen PDOs und eingerückt unter jedem PDO die zugewiesenen Objektverzeichniseinträge. In Spalte **Name** steht entweder die COB-ID des PDOs oder der Name des "gemappten" Objekts, wie er in der Gerätebeschreibung und im Objektverzeichnis verwendet wird. Außerdem sind **Index**, **Subindex** und Bitlänge (**Bitlength)** des PDOs bzw. des gemappten Objekts angegeben.

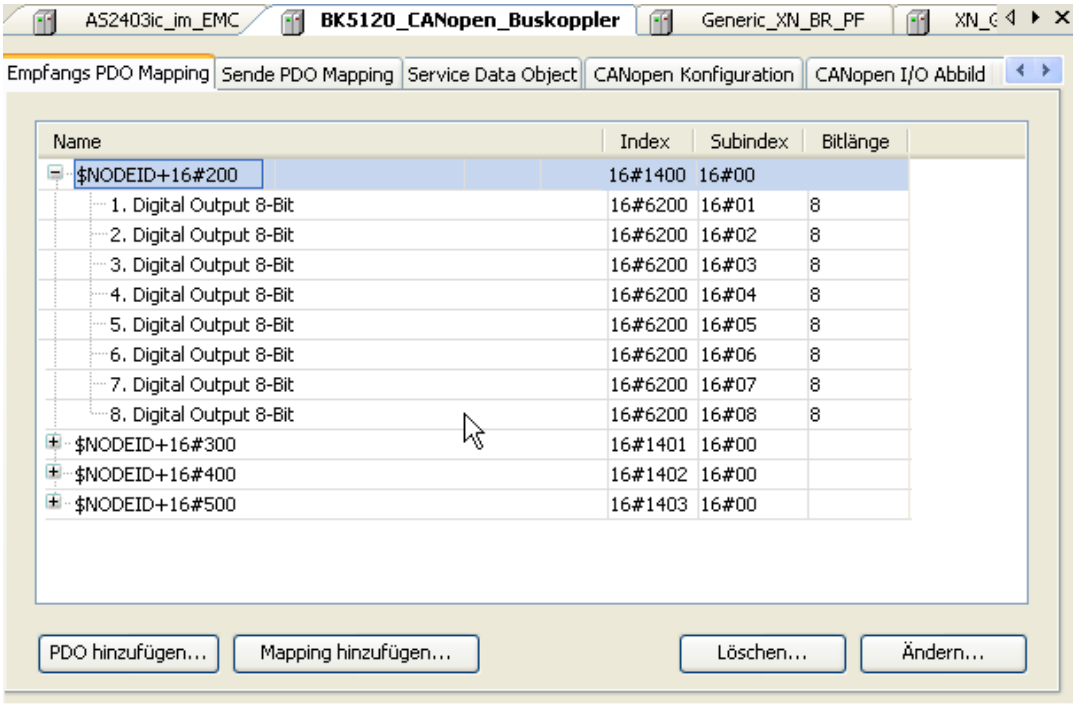


Abb. 478 Dialog 'Empfangs PDO Mapping'

Durch einen Doppelklick auf einen PDO-Eintrag öffnet sich der Dialog **PDO Eigenschaften**, der die weiter unten beschriebenen Eigenschaften anzeigt.

Wenn die Option **PDO Mapping automatisch konfigurieren** aktiviert ist ...

.... was für Standardapplikationen sehr empfohlen wird, werden die Zuordnungen, die hier im Dialog dargestellt werden, automatisch aus den Gerätebeschreibungsdateien erzeugt und können hier nicht verändert werden. In diesem Fall werden alle relevanten Parameter (z.B. digitale Ein- und Ausgänge) gemäß einer definierte Priorisierung zugewiesen.

Wenn die Option **PDO Mapping automatisch konfigurieren** nicht aktiviert ist ...

... kann das Mapping manuell konfiguriert werden. In diesem Fall können existierende PDOs bearbeitet oder neue definiert werden und mit den gewünschten Zuordnungen versehen werden. Mit Schaltfläche **Löschen...** kann ein selektiertes PDO bzw. eine einzelne Zuweisung aus der Konfiguration entfernt werden.

Bearbeiten eines bestehenden PDO Mappings

Um eine bestehende Zuweisung unterhalb eines PDOs zu bearbeiten, selektieren Sie diese in der Liste und drücken Schaltfläche **Bearbeiten**. Der Dialog **Wählen Sie einen Eintrag aus dem Objektverzeichnis** öffnet sich, wo eine neue Zuweisung hinzugefügt oder eine bestehende modifiziert werden kann. Sehen Sie weiter unten eine Beschreibung des Dialogs.

Neue PDOs und Zuordnungen anlegen

Um ein neues PDO zu definieren, verwenden Sie Schaltfläche **PDO hinzufügen**. Der Dialog **PDO Eigenschaften** öffnet sich, wo die Standardvorgaben für ein PDO entsprechend angepaßt werden können.

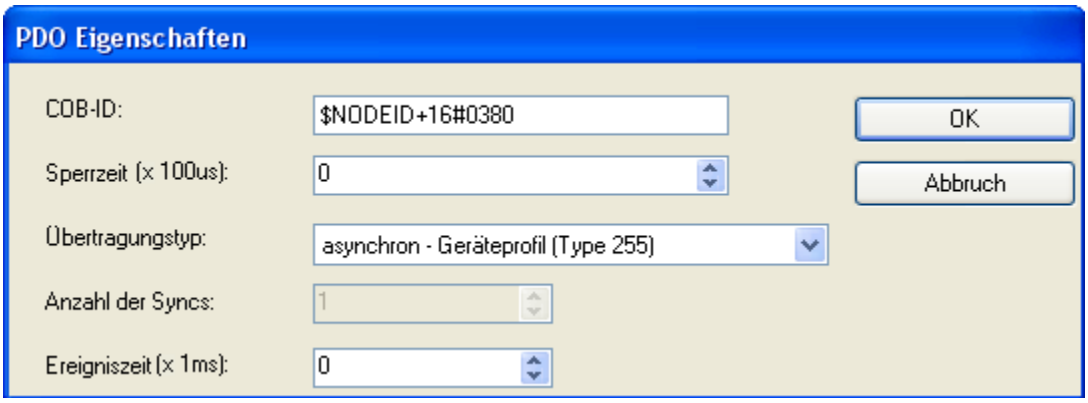


Abb. 479 PDO Eigenschaften Dialog

COB-ID: Jede PDO-Nachricht muss eine eindeutige COB-ID (Communication Object Identifier) haben. Das COB-ID Feld zeigt die Definition aus der Gerätebeschreibungsdatei (EDS-Datei). Üblicherweise wird sie basierend auf der Knotennummer des Moduls generiert, wobei "$NodeID" die aktuelle Knotennummer repräsentiert. Beispiel: "$NodeID+0x200". Die aktuelle Definition der COB-ID kann hier vom Benutzer verändert werden.

Sperrzeit (x 100µs): Dieses Feld ist nur editierbar, wenn das Gerät die Funktionalität unterstützt: ist die minimale Zeit zwischen zwei Nachrichten dieses PDOs. Die Einstellung kann verwendet werden, um zu vermeiden dass PDOs, die bei Änderung des Wertes übertragen werden, nicht zu häufig versendet werden. Die Einheit ist 100 µs. Default: "0". Mögliche Werte: 0-65535.

Übertragungstyp: Hier wird der gewünschte Übertragungsmodus eingestellt. Mögliche Einstellungen:

acyclic - synchronous: das PDO wird zwar synchron übertragen, aber nicht periodisch. (default)

cyclic – synchronous: das PDO wird synchron übertragen, wobei 'Anzahl der Syncs' (s.u.) die Anzahl der Synchronisationsnachrichten angibt, die zwischen zwei Übertragungen dieses PDO's liegen

synchronous – RTR only: das PDO wird nach einer Synchronisationsnachricht aktualisiert, aber nicht versendet. Übertragen wird es nur auf eine explizite Anfrage hin (Remote Transmission Request)

asynchronous – RTR only: das PDO wird aktualisiert und nur auf eine explizite Anfrage hin übertragen (Remote Transmission Request)

asynchronous – device profile specific and asynchronous - manufacturer specific: das PDO wird nur nach bestimmten Ereignissen übertragen.

Anzahl der Syncs: Wenn zyklische Übertragung eingestellt ist, geben Sie hier die Anzahl der Synchronisationsnachrichten an (siehe 'Zykluszeit' im CANopen Remote Device Dialog), die zwischen zwei Übertragungen des PDOs versendet werden sollen. Default: "1". Mögliche Werte: 1-240.

Ereigniszeit (x 1ms): Dieses Feld ist nur editierbar, wenn das Gerät die Funktionalität unterstützt: Wenn eine entsprechender Übertragungstyp eingestellt ist, geben Sie hier die Zeitspanne an, die zwischen zwei Übertragungen des PDOs liegen soll (in Millisekunden). Default: "0". Mögliche Werte: 0-65535.

Um **einem PDO Zuordnungen hinzufügen**, selektieren Sie den PDO-Eintrag und drücken Schaltfläche **Mapping hinzufügen...**. Der Dialog **Wählen Sie einen Eintrag aus dem Objektverzeichnis** öffnet sich, wo das gewünschte Objekt ausgewählt werden kann, das dem PDO zugeordnet werden soll.

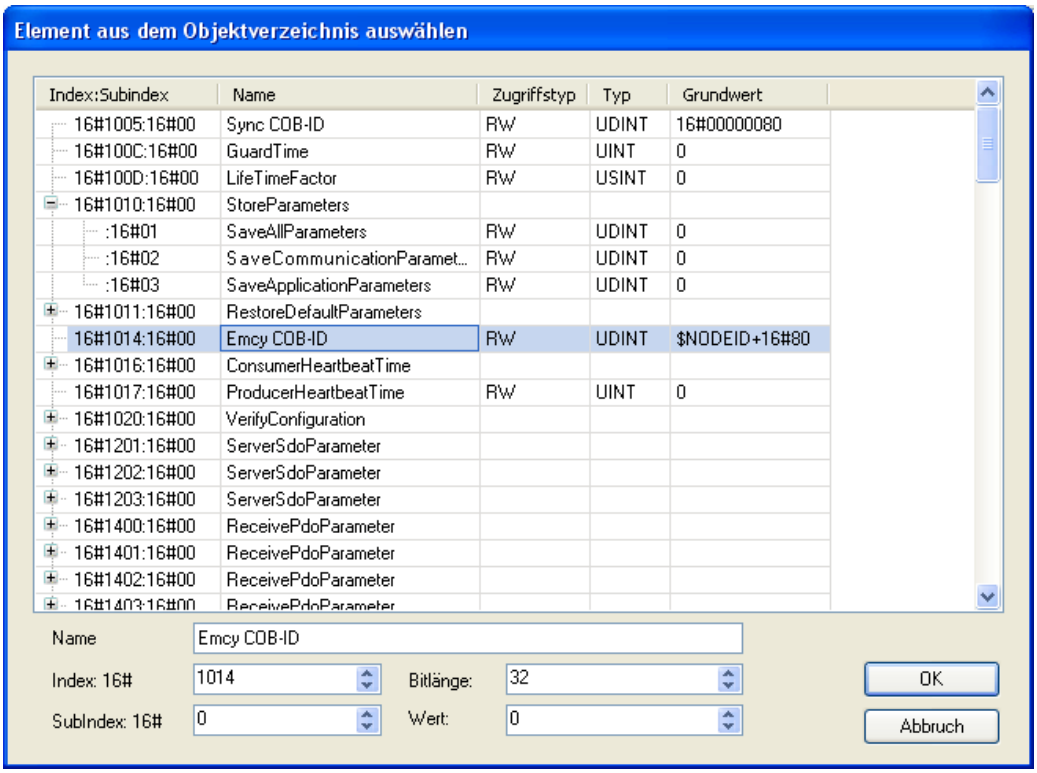


Abb. 480 Dialog 'Wählen Sie einen Eintrag aus dem Objektverzeichnis'

Die Tabelle zeigt alle zuordenbaren Objektverzeichniseinträge, die in der EDS-Datei des Geräts definiert sind. Ggfs. öffnen sie die einzelnen Subindex-Einträge mit Hilfe des Plus-Zeichens vor einem Index-Eintrag. Im Fall von Empfangs-PDOs werden hier nur **nur Objekte mit Schreibzugriff** angeboten (flag = w), im Fall von Sende-PDOs solche mit Lesezugriff. Jedes Objekt wird mit Index, Subindex, Name, Wert, Flag für den möglichen Zugriff (RW=lesbar und schreibbar, RO=nur lesbar, WO=nur schreibbar, CONST=Konstante und nur lesbar, etc.) Datentyp und Grundwert.

Für den gerade selektierten Eintrag werden in den unter der Tabelle liegenden Feldern der Name, der Index, der Subindex und die Bitlänge angezeigt und können hier **bearbeitet** werden.

Sende PDO Mapping

Dieser Dialog ist nur verfügbar, wenn die Option Enable Expert PDO Settings für das Gerät aktiviert ist.

In diesem Dialog einer CANopen Gerätekonfiguration ist das Mapping für Sende-PDOs definiert, d.h. die Zuordnung von **Eingängen** (Objektverzeichniseinträge) zu einem PDO zum Zweck des Datenaustauschs im CANbus.

Zur Bedienung sehen Sie bitte die Beschreibung des Dialogs Empfangs PDO Mapping.

Service Data Object

Während der Initialisierung des CAN-Busses werden die aktuellen Konfigurationseinstellungen mit Hilfe von Service Data Objects (SDO) zu den CAN-Modulen übertragen.

Im 'Service Data Object'-Konfigurationsdialog können die gewünschten SDOs konfiguriert werden. Es kann hier auch definiert werden, in welcher Reihenfolge sie übertragen werden sollen und was im Falle eines Übertragungsfehlers geschehen soll.

Hinweis: Wenn Option Enable Expert PDO Settings für das aktuelle Gerät nicht aktiviert ist, werden nur die benutzerdefinierten SDOs hier angezeigt.

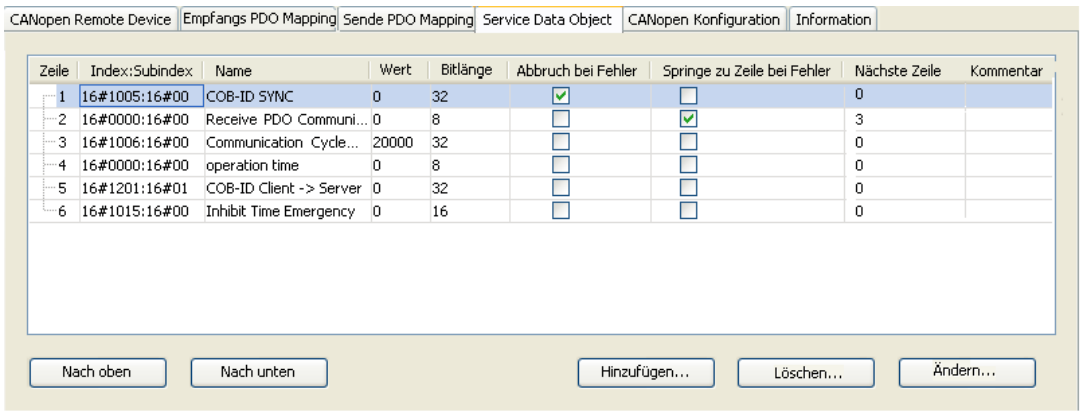


Abb. 481 Service Data Object Dialog

Um ein SDO hinzuzufügen, verwenden Sie Schaltfläche **Hinzufügen...**

Der Dialog **Wählen Sie einen Eintrag aus dem Objektverzeichnis** öffnet sich, der - wie für die Mapping-Konfiguration - die verfügbaren Objektverzeichniseinträge, wie in der EDS-Datei definiert, anbietet.

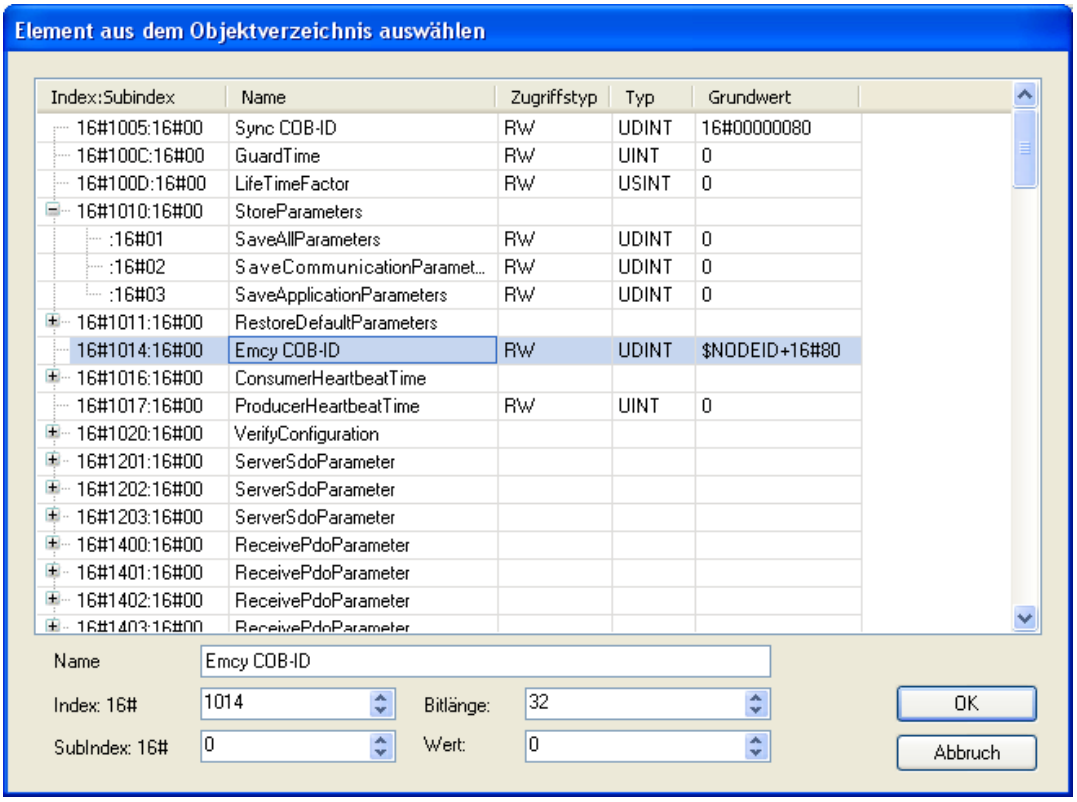


Abb. 482 Dialog 'Wählen Sie einen Eintrag aus dem Objektverzeichnis

In diesem Fall werden nur die schreibbaren Objekte angezeigt (Zugriffstyp = *W*). Bevor Sie ein SDO zur Auswahl im Service Data Object Dialog hinzufügen, können Sie seine Parameter in den Feldern unterhalb der Tabelle verändern. Somit können Sie auch - durch das Eingeben eines neuen Index/Subindex-Wertes - ein SDO anlegen, das in der EDS-Datei nicht beschrieben ist.

Die Reihenfolge (von oben nach unten) in der Tabelle der ausgewählten SDOs entspricht der **Abfolge**, in der die SDOs zum Modul übertragen werden. Mit Hilfe der Schaltflächen **Nach oben** und **Nach unten** kann der gerade selektierte Eintrag innerhalb der Tabelle verschoben werden.

Das Feld **Wert** in der Tabelle kann editiert werden. Zu diesem Zweck selektieren Sie das Feld und drücken dann die <Leertaste> oder führen einen Mausklick auf das Feld aus.

Das Verhalten im Falle eines Fehlers während des SDO Transfers kann durch Aktivieren einer der folgenden Optionen definiert werden:

Abbruch bei Fehler: Die Übertragung wird mit Fehlerstatus abgebrochen.

Springe zu Zeile bei Fehler und **Nächste Zeile**: Die Übertragung wird mit dem SDO fortgesetzt, das in der in Spalte 'Nächste Zeile' angegeben ist. Dort ist die Nummer der Zeile (Spalte '**Zeile**') anzugeben, in der das SDO steht.

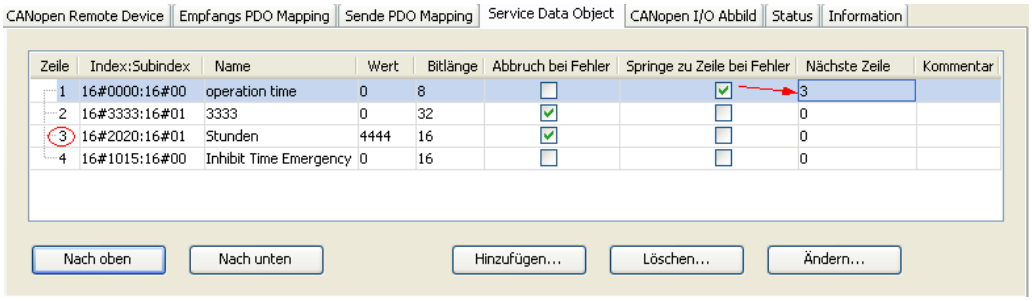


Abb. 483 Fehlerverhalten konfigurieren, Beispiel

Um die Parameter eines SDO zu verändern, das bereits in der SDO-Tabelle enthalten ist, verwenden Sie Schaltfläche **Bearbeiten...** um den Dialog **Wählen Sie einen Eintrag aus dem Objektverzeichnis** zu erhalten, in dem wie oben für das Hinzufügen eines SDOs beschrieben, gearbeitet werden kann.

Schaltfläche **Löschen...** entfernt den gerade in der Tabelle selektierten Eintrag.

CANopen Konfiguration

In diesem Dialog einer CANopen Gerätekonfiguration werden die Geräteparameter, die in der EDS-Datei stehen, aufgelistet und ihre Werte können im Feld 'Wert' bearbeitet werden. Es handelt sich um den Standard-Konfigurationsdialog eines Geräte-Editors zur Parametrierung eines Geräts.

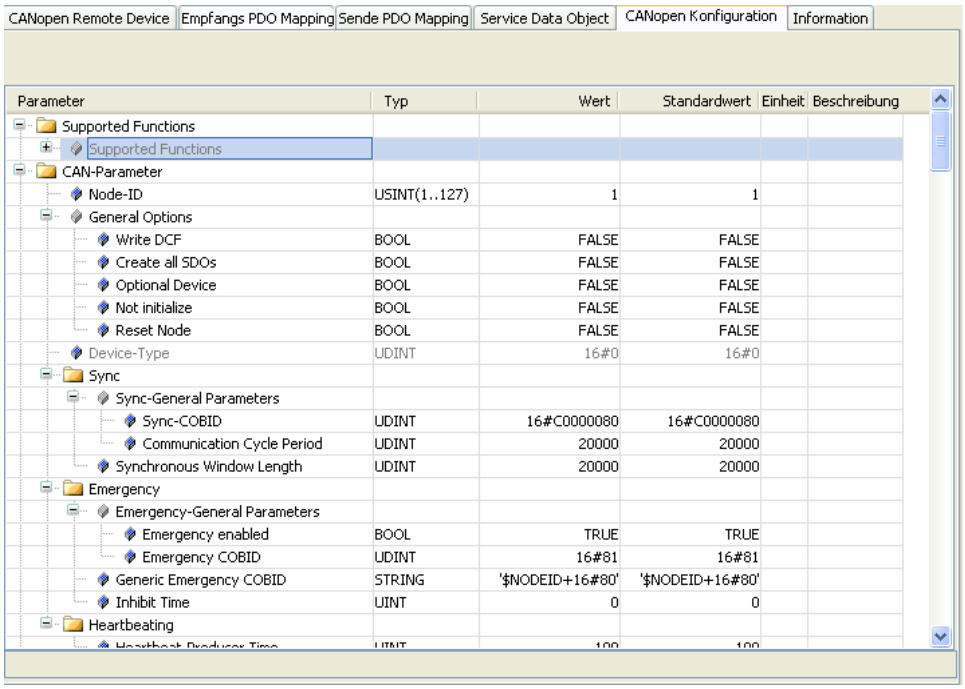


Abb. 484 Dialog CANopen Konfiguration

CANopen I/O Abbild

Dieser Dialog einer CANopen Gerätekonfiguration eines nicht-modularen Geräts dient der Konfiguration des E/A-Abbilds ("I/O Mapping") des Geräts. D.h. hier werden Projektvariablen den Ein- und Ausgangskanälen des Geräts zugewiesen. Es handelt sich um einen Standard-Konfigurationsdialog des Geräte-Editors.

Status

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Status-Informationen (z.B. "Running", "Stopped") und spezifische Diagnosemeldungen aus dem Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Information

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Informationen zum Gerät bzw. Modul, die aus der Gerätebeschreibung gelesen werden. Mögliche Angaben: Name, Hersteller, Typ, Versionsnummer, Kategorien, Bestellnummer, Abbildung.

CANopen Modul

CANopen-Modul Konfiguration

In diesem Dialog einer CANopen Konfiguration eines CANopen Moduls können die Parameter des Moduls, die aus der Modulbeschreibung entnommen sind, angezeigt und bearbeitet werden. Es handelt es sich um den Standard-Konfigurationsdialog eines Geräte-Editors zur Parametrierung eines Geräts oder Moduls. Sehen Sie bitte Geräteeditor, Konfiguration wie er zu verwenden ist.

CANopen-Modul I/O Abbild

Dieser Dialog einer CANopen Konfiguration eines CANopen Moduls dient der Konfiguration des E/A-Abbilds ("I/O Mapping") des Moduls. Es handelt sich um einen Standard-Konfigurationsdialog des Geräte-Editors.

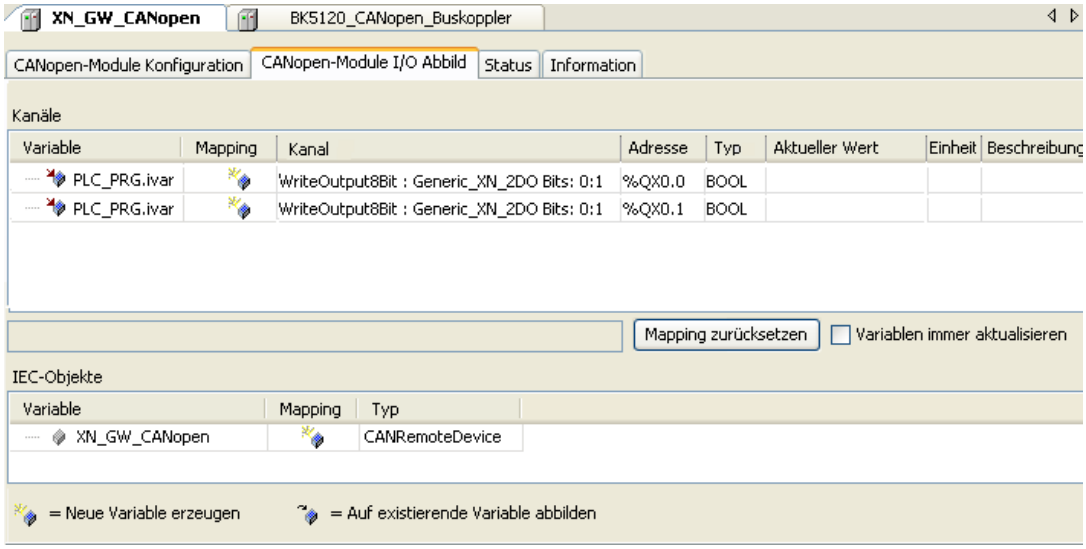


Abb. 485 Beispiel des I/O Abbild Dialogs

Wenn dieser Dialog für ein Modul verfügbar ist, ist das "Vater"-Module (CANopen Gerät) ein "modulares Gerät", das im Unterschied zu einem nicht-modularen keinen eigenen I/O Abbild Dialog hat.

Wenn die Option PDO Mapping automatisch konfigurieren aktiviert ist, wird die I/O-Abbild-Konfiguration in allen CANopen Modulen aktualisiert, sobald ein Modul unter einem modularen CANopen Gerät eingehängt oder entfernt wird!

Status

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Status-Informationen (z.B. "Running", "Stopped") und spezifische Diagnosemeldungen aus dem Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Information

Dieser Dialog einer CANopen Geräte- oder Modulkonfiguration zeigt Informationen zum Gerät bzw. Modul, die aus der Gerätebeschreibung gelesen werden. Mögliche Angaben: Name, Hersteller, Typ, Versionsnummer, Kategorien, Bestellnummer, Abbildung.

14.3 EtherCAT-Konfigurationseditor

Das Plug-In 'EtherCAT' stellt Dialoge für die Konfiguration von EtherCAT Master und Slave Modulen bereit, die im Gerätebaum eines Projekts repräsentiert sind. Diese Dialoge werden im Geräte-Editor zusätzlich zu den Standarddialogen angezeigt.

Die Konfiguration von EtherCAT-Modulen basiert auf den Gerätebeschreibungsdateien für die verwendeten Master- und Slave-Geräte und kann im Projekt in den Konfigurationsdialogen durch den Anwender noch angepaßt werden. Allerdings ist es für Standard-Applikationen ausreichend und zum Zwecke einer möglichst einfachen und fehlerfreien Verwendung ratsam, den Modus 'Automatische Konfiguration' zu aktivieren, so dass der Großteil der Konfigurationseinstellungen automatisch vorgenommen wird.

Beachten Sie die Möglichkeit, die Konfiguration von Master und Slaves auch im laufenden Betrieb zu beeinflussen, indem über die Applikation auf Master- und Slave-Instanzen zugegriffen wird, bzw. Bausteine der EtherCAT-Bibliothek verwendet werden.

Allgemeines zu EtherCAT

EtherCAT, das offene Echtzeit-Ethernet-Netzwerk, erlaubt Echtzeitleistungen, da es mit Hilfe von Twisted Pair Kuper - oder Fiberglas-Leitungen 1000 verteilte I/Os in 30 µs oder 100 Achsen pro 100 µs aktualisieren kann. Linien- Baum- oder Stern-Topologie wird dabei unterstützt. Es gibt einen Bus-Master und Slaves.

Ein Ethernet Datenpaket muss nicht mehr an jedem Knoten zunächst empfangen, dann interpretiert und die Prozessdaten weiter kopiert werden, sondern es kann praktisch ohne Zeitverlust weitergereicht werden. Die FMMU (fieldbus memory management unit) in jedem Slave-Knoten erkennt ein an ihn gerichtetes Telegramm und entnimmt die Daten, während das Telegramm bereits an das nächste Gerät weitergereicht wird. Ebenso werden Eingangsdaten dem durchlaufenden Telegramm hinzugefügt, das dabei nur um wenige Nanosekunden verzögert wird. Die Adressierung der Ethernet Terminals kann in beliebiger Reihenfolge vorliegen, da die Datenabfolge unabhängig von der physikalischen Anordnung ist. Broadcasting, Multicasting u Kommunikation zwischen Slaves ist möglich.

Exakte Synchronisation, die beispielsweise in Applikationen benötigt wird, in denen mehrere Hilfsachsen gleichzeitig koordinierte Bewegungen ausführen, ist durch akkuraten Zeitabgleich von "distributed clocks" (verteilte Uhren) möglich. Diese Technik erlaubt im Unterschied zu einer voll synchronisierten Kommunikation ein hohes Maß an Toleranz gegenüber möglichen fehlerbedingten Verzögerungen.

Installieren und Einfügen von EtherCAT Geräten

Die Kombination von EtherCAT Geräten mit einer »PLC Designer« SP Win V3 erfordert die Installation der Programmbibliothek WinPCap (freier Download beispielsweise unter winpcap.org).

Des weiteren muss die »PLC Designer« Konfigurationsdatei (...\GatewayPLC\PLC Designer.cfg) um einen der folgenden Einträge ergänzt werden:

- Component.<subsequent number>=SysEthernet
- Component.<subsequent number>=CmpEt100Drv (bei Verwendung von IntelPro100)
- Component.<subsequent number>=CmpRTL81x9Mpd (bei Verwendung von RTL8139)
- Component.<subsequent number>=CmpRTL8169Mpd (bei Verwendung von Realtek RTL8169 oder RTL8168(PCIe Version))

Um EtherCAT Geräte als Objekte im Gerätebaum des Projekts einfügen und konfigurieren zu können, müssen die Master- und Slave-Module zunächst auf dem lokalen System installiert werden. Dies erfolgt automatisch durch die Standard-Installation von »PLC Designer«, und muss ansonsten explizit mit Hilfe des Geräte-Repository Dialogs durchgeführt werden, basierend auf den **Gerätebeschreibungsdateien**, die mit der Hardware geliefert werden. Die Gerätebeschreibungsdatei für den EtherCAT Master *.devdesc.xml) legt fest, welche Slaves eingefügt werden können. Die Beschreibungen der Slaves sind ebenfalls als xml-Dateien verfügbar: Dateityp: EtherCAT XML Device Description Configuration Files).

Ein EtherCAT Master kann im Gerätebaum mit Befehl 'Gerät hinzufügen' oder 'Objekt hinzufügen' eingehängt werden. Unterhalb eines Masters können dann ein oder mehrere Slaves eingefügt werden.

Sobald ein Master-Gerät im Gerätebaum eingehängt wurde, ist die benötigte Bibliothek IoDrvEtherCAT.library automatisch im Bibliotheksverwalter eingebunden.

Zugleich wird eine EtherCAT_Master T in die Taskkonfiguration der aktuellen Applikation eingefügt. Wie üblich kann in der Konfiguration dieser Task der Aufruf einer POU hinzugefügt werden, innerhalb derer dann EtherCAT spezifische boolesche Variablen gesetzt werden können um das Verhalten der ETherCAT Konfiguration im Rahmen der Applikation zu beeinflussen:

- Unterstützung optionaler Geräte
  Das Fehlen von in der Applikation konfigurierten EtherCAT Geräten wird beim Start des Feldbusses einen Fehler erzeugen, der ein Laden des Stacks verhindert. Nach Setzen der Variablen
  <instance name of Ethercat master>.StartConfigWithLessDevice := TRUE;
  am Anfang der ersten PLC Zyklus werden fehlende Geräte als optionale Geräte behandelt, welche einen korrekten Startprozess des Busses nicht länger behindern.

- Unterdrücken von zusätzlichem Nachrichtenversand
  Um die Ausgänge so schnell wie möglich zu aktualisieren sendet der EtherCat Master für jede einzelne Task eine eigene Nachricht. Wenn nun aber beispielsweise ein angehängter Antrieb mit den Echtzeit-Ausgangsdaten synchronisiert, so sollte die Bus Zyklustask die einzige sein, der es erlaubt ist die Ausgänge zu setzen, da zusätzliche Nachrichten die Synchronisation stören würden. Um zusätzliche Task Nachrichten zu unterdrücken, muss einmalig im ersten PLC Zyklus
  <instance name of Ethercat master>.EnableTaskOutputMessage := FALSE;
  gesetzt werden.

Konfigurationsdialoge

Standardmäßig kann der Geräte-Editor für die **EtherCAT Konfiguration** mit einem Doppelklick auf ein EtherCAT-Objekt im Gerätebaum geöffnet werden, oder mit dem Befehl 'Objekt bearbeiten' wenn im Baum ein Master- oder Slave-Objekt selektiert ist.

Beachten Sie die Geräteeditor-Optionen für allgemeine Editoreinstellungen: Beispielsweise können generische Konfigurationsdialoge ausgeblendet werden.

Beachten Sie außerdem, dass die **Buszyklus-Task** in den SPS-Einstellungen eingestellt wird.

Ein EtherCAT-Konfigurationsdialog ist immer mit dem Namen des jeweiligen Master- oder Slave-Geräts betitelt und kann Registerblätter mit den folgenden Dialogen enthalten:

Für einen EtherCAT Master:

- Master
- EtherCAT Konfiguration / Master
- EtherCAT I/O Abbild / Master
- Status
- Information

Für einen EtherCAT Slave:

- Slave
- FMMU/Sync
- Expertenmodus Prozessdaten
- Prozessdaten
- Startparameter
- EtherCAT Konfiguration / Slave
- EtherCAT I/O Abbild / Slave
- Status
- Information

Zugriff auf die EtherCAT Konfiguration durch die Applikation

Für Master und Slave werden automatisch Instanzen angelegt, die im Applikationsprogramm angesprochen werden können. Beispielsweise kann so aus der Applikation ein Neustart, ein Stop oder eine Statusprüfung des EtherCAT Geräts durchgeführt werden.

- Implizite Instanz des Masters
- Implizite Instanz des Slaves
- Verkettete Liste zur Überprüfung aller Slaves

Weiterhin bietet die EtherCAT-Bibliothek Funktionsblöcke für das Lesen und Schreiben einzelner Parameter, auch während des laufenden Busbetriebs.

- Funktionsbausteine für "CAN over EtherCAT"

EtherCAT Master

Master

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Masters enthält die Grundeinstellungen für den Master, wie sie in der Gerätebeschreibungsdatei definiert sind.

EtherCAT Master benötigen keinen expliziten Kommunikationsprozessor und können ohne eine spezielle aktive Einsteckkarte implementiert werden, einfach durch Verwenden einer passiven NIC-Karte oder den "On-Board Ethernet MAC" (Media Access Controller).

Das Einfügen eines EtherCAT Master Geräts bringt automatisch das Einfügen eines EtherCAT_MasterTask in die Taskkonfiguration der aktuell aktiven Applikation mit sich.

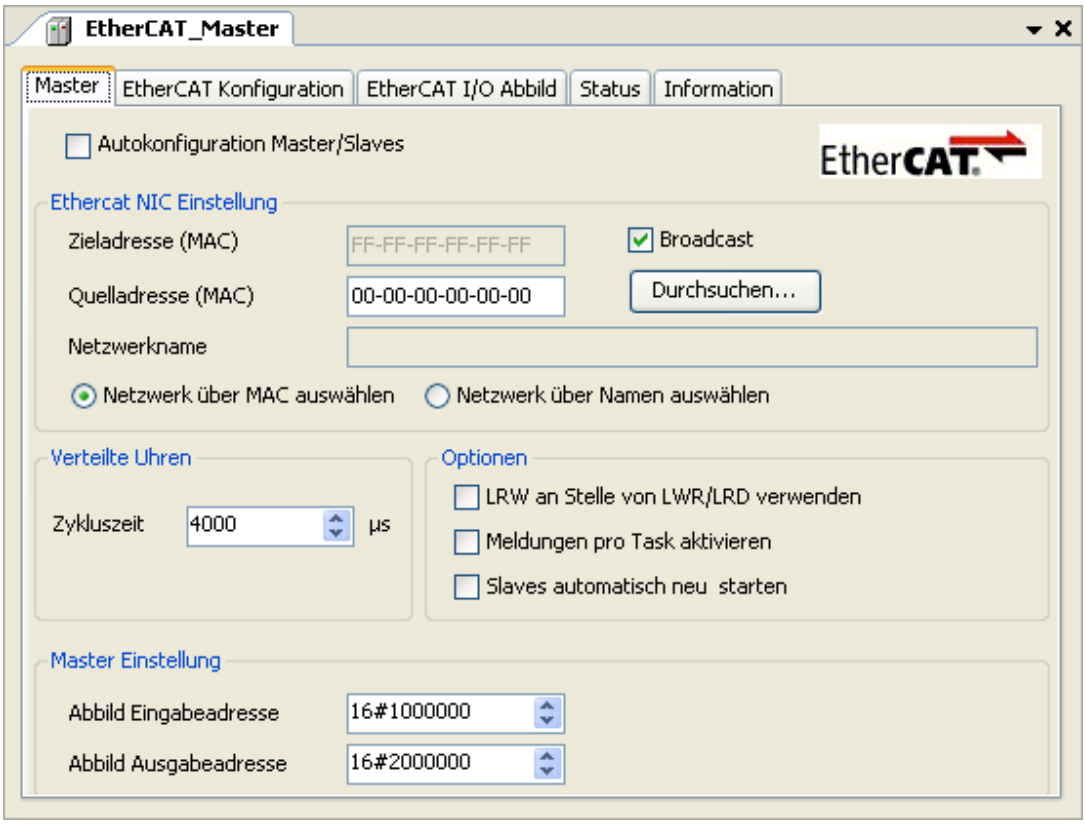


Abb. 486 EtherCAT Master Konfigurationsdialog

Automatische Konfiguration Master/Slaves: Wenn diese Option aktiviert ist, wird der Großteil der Master- und Slave-Konfiguration automatisch durchgeführt, basierend auf den Gerätebeschreibungsdateien und impliziten Berechnungen. In diesem Fall ist der Dialog für die FMMU/Sync-Einstellungen nicht verfügbar.

Es kann jedoch auch bei aktivierter Autokonfiguration des Masters für jeden einzelnen Slave explizit ein "Expertenmodus" aktiviert werden, der die manuelle Bearbeitung der automatisch generierten Prozessdatenkonfiguration erlaubt.

Hinweis: Der Autokonfigurationsmodus ist standardmäßig aktiviert und für Standardapplikationen ausreichend und sehr empfohlen. Wenn der Modus nicht aktiviert ist, müssen alle Konfigurationseinstellungen für Master und Slave(s) manuell vorgenommen werden und Expertenkenntnisse sind erforderlich! Zur Konfiguration von Slave-zu-Slave-Kommunikation muss der Autokonfigurationsdialog allerdings ausgeschaltet werden.

EtherCAT NIC Setting

Zieladresse (MAC): MAC-Adresse des EtherCAT Geräts. Wenn die Option **Broadcast** aktiviert ist, muss keine Adresse angegeben werden.

Quelladresse (MAC): MAC-Adresse der Steuerung (Zielsystem). Die Schaltfläche **Durchsuchen** hilft beim Setzen der Adresse eines der verfügbaren Zielsysteme.

Verteilte Uhren ("Distributed Clocks")

Zykluszeit [µs]: Zeitspanne, nach der ein neues Datentelegramm auf dem Bus versendet wird. Wenn die "Distributed Clock" Funktionalität im Slave aktiviert ist, wird die hier angegebene Master-Zykluszeit zu den Slave-Uhren übertragen. So kann eine genaue Synchronisation des Datenaustausches erreicht werden, die vor allem wichtig ist, wenn räumlich verteilte Prozesse gleichzeitige Aktionen erfordern (z.B. in Applikationen, in denen mehrere Achsen gleichzeitig koordinierte Bewegungen ausführen müssen). Eine sehr präzise, netzwerkweite Zeitbasis mit einem Jitter von wesentlich weniger als 1 Mikrosekunde kann auf diese Art erreicht werden.

Diagnose Nachricht: Im Online-Modus wird das Hochfahren des Systems überwacht und entsprechende Diagnosenachrichten werden ausgegeben. Wenn die Meldung "All slaves done" erscheint, ist die Konfiguration beendet und der Bus ist in Betrieb.

Options

LRW anstelle von LWR/LRD verwenden : Wenn diese Option aktiviert ist, ist direkte Kommunikation von Slave zu Slave möglich. Anstelle von separaten Lese- (LRD) und Schreibbefehlen (LWR), werden kombinierte read/write Befehle (LRW) verwendet.

Meldungen pro Task aktivieren : Wenn diese Option aktiviert ist, können Lese- und Schreibbefehle, also die Handhabung der Eingangs- und Ausgangsmeldungen mit verschiedenen Tasks gesteuert werden.

Slaves automatisch neu starten: Wenn diese Option aktiviert ist, versucht der Master bei einem Kommunikationsabbruch sofort die Slaves neu zu starten.

Master Einstellungen

Diese Einstellungen können nur bearbeitet werden, wenn der Modus 'Automatische Konfiguration' deaktiviert ist (s.o.), ansonsten werden sie automatisch vorgenommen und sind hier im Dialog gar nicht sichtbar.

Eingangsdaten Adresse: Erste logische Adresse des ersten Slaves für Eingangsdaten.

Ausgangsdaten Adresse: Erste logische Adresse des ersten Slaves für Ausgangsdaten.

EtherCAT Konfiguration / Master

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Masters enthält die Master-Parameter, die in der Gerätebeschreibungsdatei definiert sind.

Wenn der Autokonfigurationsmodus aktiviert ist, werden die Parameter hier automatisch gemäß den Angaben aus der Beschreibungsdatei und der Netzwerk-Topologie eingestellt. Für Standardapplikationen ist es normalerweise dann auch **nicht erforderlich, sie zu bearbeiten**.

Die Parameter werden mit **Parameter**namen, Daten**Typ**, aktuell eingestelltem **Wert**, **Standardwert**, **Einheit** und einer kurzen **Beschreibung** dargestellt.. Das Feld 'Wert' kann editiert werden, was aber nur wirksam wird, wenn der Autokonfigurationsmodus deaktiviert ist!

EtherCAT I/O Abbild / Master

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Masters ermöglicht, Projektvariablen auf die EtherCAT Master Ausgänge und Eingänge abzubilden. Somit kann das Steuerungsgerät am EtherCAT Master durch das Applikationsprogramm kontrolliert werden.

Wie das "Mapping" durchgeführt wird, sehen Sie bitte in der Beschreibung des allgemeinen I/O-Abbild-Dialogs im Geräteeditor.

Die automatisch angelegte Instanz des Masters, die in der Applikation verwendet werden kann, wird unten im Dialog (IEC-Objekte) angezeigt.

Status

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Geräts zeigt Status-Informationen (z.B. "Running", "Stopped") und spezifische Diagnosemeldungen aus dem Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Information

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Masters zeigt folgende allgemeine Informationen zum Gerät an, soweit sie in der Gerätebeschreibungsdatei vorliegen: Name, Hersteller, Versionsnummer, Kategorien, Bestellnummer, Beschreibung, Abbildung.

EtherCAT Slave

Slave

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Slaves zeigt die Grundeinstellungen für den Master, wie sie in der Gerätebeschreibungsdatei definiert sind.

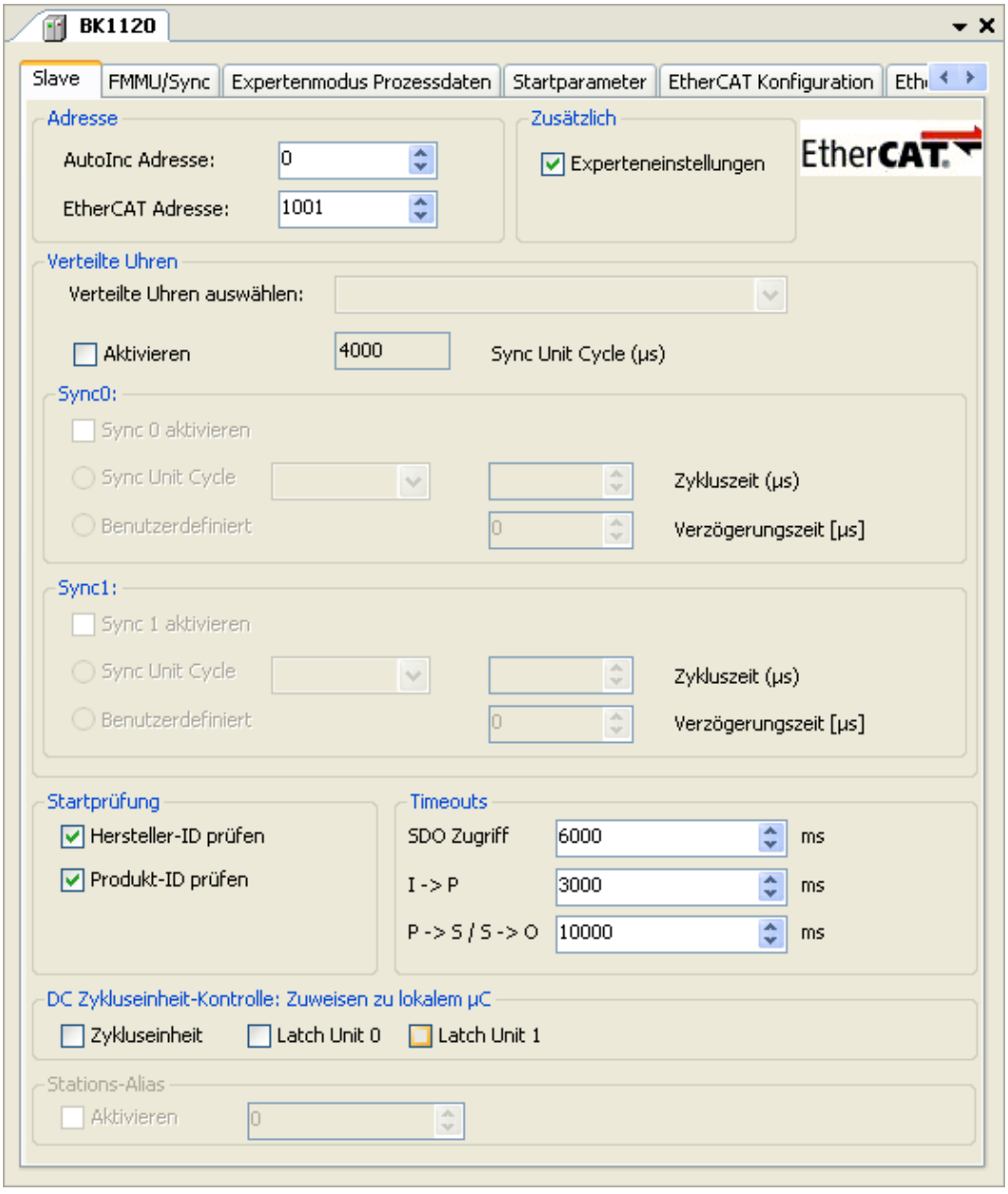


Abb. 487 EtherCAT Slave Konfigurationsdialog

Adresse

Die folgenden Felder können nur bearbeitet werden, wenn der Autokonfigurations-Modus (im Master) nicht aktiviert ist:

AutoInc Adresse: Selbstinkrementierende Adresse (16-bit), die sich aus der Position des Slaves im Netzwerk ergibt. Die Adresse wird nur beim Hochfahren des Systems verwendet, wenn der Master seinen Slaves die EtherCAT Adressen zuweist. Wenn zu diesem Zweck das erste Telegramm durch die Slaves läuft, erhöht jeder durchlaufene Slave seine AutoInc Adresse um 1. Der Slave mit Adresse 0 erhält schließlich die Daten. Eine mögliche Eingaben hier ist z.B. "-8".

EtherCAT Adresse: Endgültige Adresse des Slaves, durch den Master beim Hochfahren zugewiesen. Diese Adresse ist unabhängig von der Position des Slaves im Netzwerk.

Verteilte Uhren ("Distributed Clock")

DC auswählen: Das Aufklappmenü enthält alle in der Gerätebeschreibung vorgesehenen Einstellungen für verteilte Uhren.

aktivieren: Wenn die Funktionalität "Verteilte Uhren" aktiviert ist, wird die Zykluszeit für den Datenaustausch, angezeigt im Feld **Sync Zykluszeit (µs)**, durch die Zykluszeit des Masters bestimmt. Somit kann die Master-Uhr den Datenaustausch im Netzwerk synchronisieren.

Die Einstellungen für das Handhaben der Synchronisationseinheit(en) sind slave-abhängig:

Sync0:

Sync 0 aktivieren: Wenn diese Option aktiviert ist, wird die Synchronisationseinheit "Sync0" (Beckhoff) verwendet. Eine Synchronisationseinheit beschreibt einen Satz von Prozessdaten, der synchron ausgetauscht wird.

Sync Zykluszeit: Wenn diese Option aktiviert ist, wird die Master Zykluszeit, multipliziert mit dem gewählten Faktor, als Synchronisations-Zykluszeit für den Slave verwendet. Das Feld **Zykluszeit (µs)** zeigt die aktuell eingestellte Zykluszeit.

Benutzerdefiniert: Wenn diese Option aktiviert ist, kann die gewünschte Zykluszeit in Mikrosekunden im Feld **Zykluszeit (µs)** eingetragen werden.

Sync1:

Sync 1 aktivieren: Wenn diese Option aktiviert ist, wird die Synchronisationseinheit "Sync1" (Beckhoff) verwendet. Eine Synchronisationseinheit beschreibt einen Satz von Prozessdaten, der synchron ausgetauscht wird.

Sync Zykluszeit: Wenn diese Option aktiviert ist, wird die Master Zykluszeit, multipliziert mit dem gewählten Faktor, als Synchronisations-Zykluszeit für den Slave verwendet. Das Feld **Zykluszeit (µs)** zeigt die aktuell eingestellte Zykluszeit.

Benutzerdefiniert: Wenn diese Option aktiviert ist, kann die gewünschte Zykluszeit in Mikrosekunden im Feld **Zykluszeit (µs)** eingetragen werden.

Diagnose

Dieser Abschnitt erscheint nur im Online Modus.

Aktueller Zustand : Der Zustand des Slaves wird angezeigt. Mögliche Angaben: Init, Pre-Operational, Safe Operational, Operational.

Wenn Status "Operational" angezeigt wird, ist die Slave-Konfiguration korrekt abgeschlossen.

Erweiterte Einstellungen

Aktiviere Experteneinstellungen: Wenn diese Option aktiviert ist, können zusätzlich Einstellungen für eine Startüberprüfung und Zeitüberwachung vorgenommen werden (s.u.), und das Registerblatt Expertenmodus Prozessdaten ist zusätzlich im Geräte-Editor verfügbar. Bedenken Sie allerdings, dass für Standardapplikationen der Autokonfigurationsmodus zu empfehlen und völlig ausreichend ist, so dass "Experteneinstellungen" nicht erforderlich sind.

Startüberprüfung

Standardmäßig werden beim Hochfahren des Systems die **Hersteller ID** und/oder die **Produkt ID** des Geräts gegen die aktuell vorliegenden Konfigurationseinstellungen geprüft. Wenn sie nicht übereinstimmen, wird der Bus gestoppt und keine weiteren Aktionen ausgeführt. Dies dient dazu, das Laden einer fehlerhaften Konfiguration auf das Bussystem zu verhindern.

Hier können Sie die entsprechenden Optionen deaktivieren, um die Prüfung auszuschalten.

Zeitüberwachung

Standardmäßig ist keine Zeitüberwachung für die folgenden Aktionen definiert. Bei Bedarf kann jedoch hier eine entsprechende "Timeout"-Zeit angegeben werden (in Mikrosekunden):

SDO Zugriffe: Senden der SDO-Liste beim Systemstart.

I -> P: Weiterschalten von Modus "Init" in Modus "Preoperational".

P -> S / S-> O: Weiterschalten von Modus "Preoperational" in Modus "Safe Operational" bzw. von "Safe Operational" in "Operational".

DC Zykluseinheit-Kontrolle: Zuweisen zu lokalem µC

Hier kann eine oder mehrere Optionen bezüglich der "Verteilte Uhren"-Funktionalität aktiviert werden, die auf dem lokalen Mikroprozessor verwendet werden soll(en). Die Kontrolle erfolgt in Register 0x980 im EtherCAT Slave: Die möglichen Einstellungen: Zykluseinheit (cyclic unit), Latch Unit 0, Latch Unit 1.

Stations-Alias

Nur wenn es in der Gerätebeschreibung des Slaves unterstützt wird, ist diese Einstellung aktivierbar. Mit ihr kann direkt eine Alias-Adresse zugewiesen werden, um die Adresse des Slaves unabhängig von seiner Position im Bus zu machen.

FMMU/Sync

Dieses Registerblatt ist nur dann im Geräte-Editor eines EtherCAT Slaves verfügbar, wenn der AutoKonfigurationsmodus im **Master** deaktiviert ist. Es zeigt die FMMUs und Sync Managers des Slaves, wie sie in der Gerätebeschreibungsdatei definiert sind, und bietet die Möglichkeit, diese zu bearbeiten, beispielsweise zur Konfiguration einer Slave-zu-Slave Kommunikation.

Hinweis: Beachten Sie, dass es sich um Experteneinstellungen handelt, die für Standardapplikationen in der Regel nicht erforderlich sind!

FMMU:

Diese Tabelle zeigt die "Fieldbus Memory Management Units" des Slaves, die für das Handhaben der Prozessdaten verwendet werden. Es ist jeweils die Zuordnung der logischen Adresse (**Global StartAdr.**) auf eine physikalische Adresse (**Phys StartAdr.**) definiert. Bitweises "Mapping" ist möglich. Für das Hinzufügen neuer Einheiten bzw. das Bearbeiten vorhandener kann der Dialog **Bearbeite FMMU** über Schaltfläche **Hinzufügen...**bzw. **Bearbeiten...** geöffnet werden.

Sync Manager

Diese Tabelle zeigt den/die **Synchronisations-Manager** des Slaves. Für jeden verfügbaren Sync Manager **Typ** (Mailbox In, Mailbox Out, Inputs, Outputs) sind u.a. die physikalische **Start Adresse**, der **Zugriffs**typ, der **Puffer** und die physikalische Adresse, an die die **Interrupts** gesendet werden sollen, definiert. Neue Sync Manager können hinzugefügt werden und bestehende können bearbeitet werden. Dazu dient der Dialog **Bearbeite SyncMan**, der über die Schaltflächen **Hinzufügen** bzw. **Bearbeiten** geöffnet werden kann.

Expertenmodus Prozessdaten

Dieses Registerblatt ist nur dann im Geräte-Editor eines EtherCAT Slaves verfügbar, wenn die Experteneinstellungen für den Slave aktiviert sind. Es bietet eine andere und detailliertere Ansicht der Prozessdaten, die auch im Registerblatt Prozessdaten dargestellt sind. Zusätzlich kann hier der Download der **PDO Zuordnung** und der **PDO Konfiguration** aktiviert werden.

Prozessdaten

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Slaves zeigt die **Prozessdaten** für die Ein- und Ausgänge des Slaves, jeweils definiert durch Namen, Typ und Index aus der Gerätebeschreibungsdatei .

Den hier ausgewählten Eingängen (zu lesen) und Ausgängen (zu schreiben) des Geräts können dann im Dialog I/O Abbild Projektvariablen zugewiesen werden.

Die aktuelle Auswahl, die hier dargestellt wird, kann verändert werden, indem über einen Mausklick auf das Kästchen vor einem Eintrag ein Haken gesetzt bzw. entfernt wird und somit das jeweilige Datum der Auswahl hinzugefügt oder daraus entfernt wird.

Startparameter

Dieser Dialog steht nur dann auf einem Registerblatt des EtherCAT Slave Konfigurationsdialogs bereit, wenn der Modus Expertenmodus für den Slave aktiviert ist.

Hier können für den aktuellen Slave SDOs definiert werden, die spezifizierte Parameter beim Systemstart zum Gerät übermitteln. Der Dialog steht nur bereit, wenn das Gerät "CAN over EtherCAT" unterstützt. Das Objektverzeichnis,das die erforderlichen Datenobjekte enthält, ist in der EtherCAT XML-Beschreibungsdatei beschrieben, oder in einer EDS-Datei, die in der XML-Datei referenziert wird.

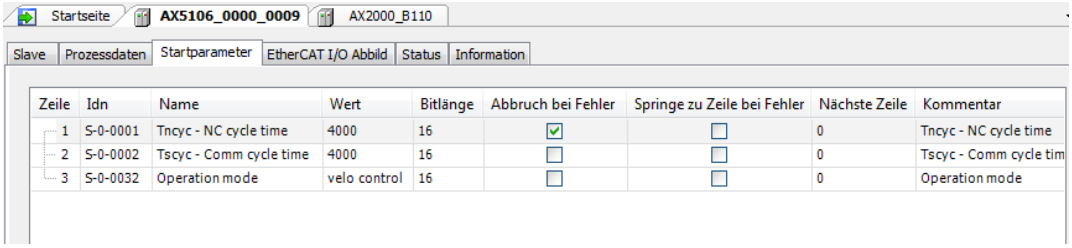


Abb. 488 Startparameter Dialog

CAN over EtherCAT

Um einen SDO Eintrag in der Konfigurationstabelle einzufügen, verwenden Sie Schaltfläche **Hinzufügen...**

Der Dialog **Wählen Sie einen Eintrag aus dem Objektverzeichnis** öffnet sich, in dem die verfügbaren Objektverzeichniseinträge, wie in der EDS-Datei definiert, aufgelistet werden.

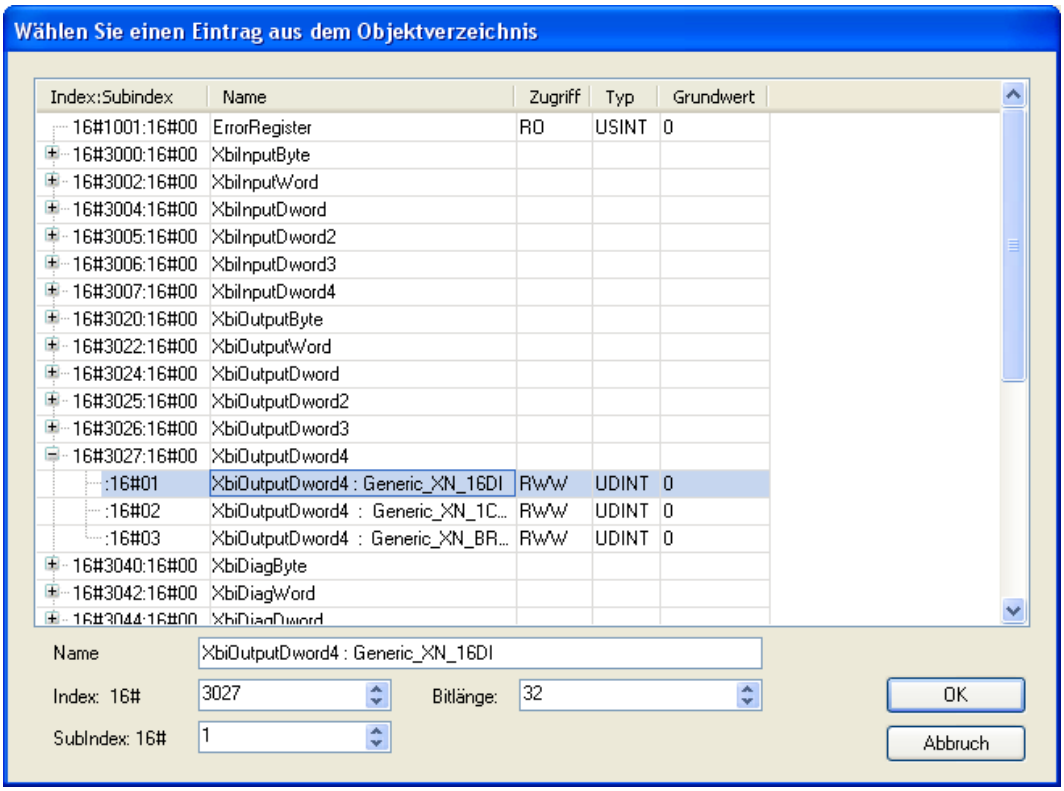


Abb. 489 Dialog 'Wählen Sie einen Eintrag aus dem Objektverzeichnis'

Bevor ein SDO der Konfiguration hinzugefügt wird, können seine Parameter in den Eingabefeldern unterhalb der Objekttabelle geändert werden. Durch Angeben von neuen Index/Subindex-Einträgen kann dem SDO ein neues Objekt hinzugefügt werden, das noch nicht in der EDS-Datei beschrieben ist. Dies ist nützlich, wenn nur ein unvollständiges oder noch gar kein Objektverzeichnis vorhanden ist.

Die Reihenfolge (von oben nach unten) in der SDO-Tabelle gibt die **Reihenfolge** vor, in der die SDOs zum Modul übertragen werden. Mit den Schaltflächen **Nach oben** und **Nach unten** kann der gerade selektierte Eintrag innerhalb der Liste verschoben werden.

Die Felder **Grundwert** und **Kommentar** können ebenfalls direkt editiert werden. Dazu wählen Sie das Feld aus und drücken Sie die <Leertaste> oder klicken ein weiteres mal auf das Feld.

Das Verhalten im Falle eines Fehlers während des SDO-Transfers kann durch Aktivieren einer der folgenden Optionen definiert werden:

Abbruch bei Fehler: Der Transfer wird mit Error-Status abgebrochen.

Springe zu Zeile bei Fehler und **Nächste Zeile**: Der Transfer wird mit dem SDO fortgesetzt, das in der in 'Next line' angegebenen Zeile steht (die Zeilennummer ist angegeben; Zeilennummern siehe Spalte **Zeile**)

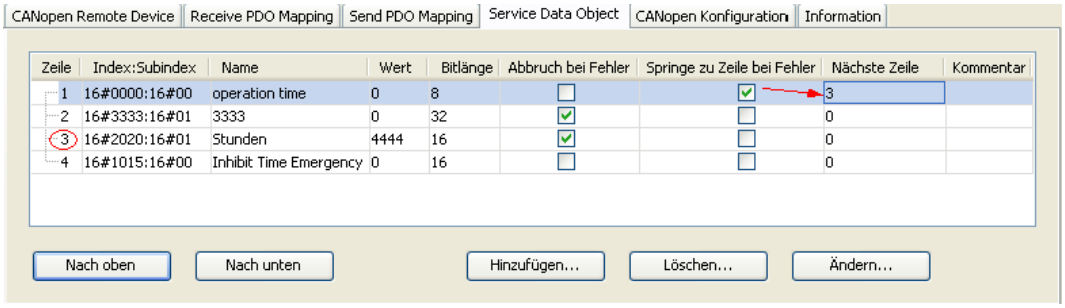


Abb. 490 Fehler-Handling konfigurieren, Beispiel

Um die Parameter eines SDOs zu verändern, das bereits in der SDO-Konfigurationstabelle enthalten ist, verwenden Sie Schaltfläche **Bearbeiten...** um den Dialog **Wählen Sie einen Eintrag aus dem Objektverzeichnis** zu erhalten, in dem Sie arbeiten können wie es oben für das Hinzufügen eines SDOs beschrieben ist.

Schaltfläche **Löschen...** entfernt den gerade selektierten Eintrag.

Servodrive over EtherCAT

Um einen IDN Eintrag in der Konfigurationstabelle einzufügen, verwenden Sie Schaltfläche Hinzufügen...

Der Dialog Wählen Sie einen Eintrag aus dem Objektverzeichnis öffnet sich, in dem die verfügbaren Objektverzeichniseinträge, wie in der XML-Datei definiert, aufgelistet werden.

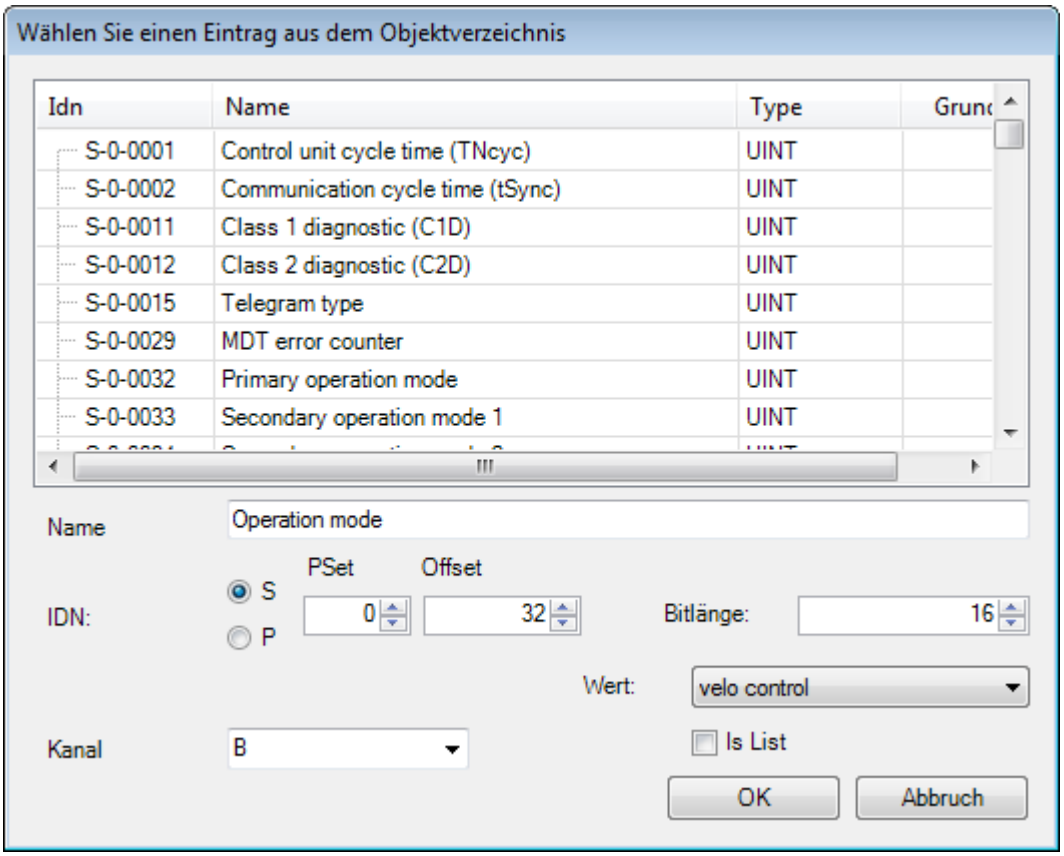


Abb. 491 Dialog 'Wählen Sie einen Eintrag aus dem Objektverzeichnis'

Bevor ein IDN der Konfiguration hinzugefügt wird, können seine Parameter in den Eingabefeldern unterhalb der Objekttabelle geändert werden. Durch Angeben von neuen PSet/Offset-Einträgen kann dem IDN ein neues Objekt hinzugefügt werden, das noch nicht in der XML-Datei beschrieben ist. Dies ist nützlich, wenn nur ein unvollständiges oder noch gar kein Objektverzeichnis vorhanden ist.

Das Auswahlfeld Kanal wird automatisch angezeigt, wenn das Objekt mehrere Unterobjekte hat. Beispiel: Ein Servoregler kann zwei Servomotoren ansteuern.

Die Reihenfolge (von oben nach unten) in der IDN-Tabelle gibt die Reihenfolge vor, in der die IDNs zum Modul übertragen werden. Mit den Schaltflächen [Nach oben] und [Nach unten] kann der gerade selektierte Eintrag innerhalb der Liste verschoben werden.

Die Felder Grundwert und Kommentar können ebenfalls direkt editiert werden. Dazu wählen Sie das Feld aus und drücken Sie die [Leertaste] oder klicken ein weiteres mal auf das Feld.

Das Verhalten im Falle eines Fehlers während des IDN-Transfers kann durch Aktivieren einer der folgenden Optionen definiert werden:

Abbruch bei Fehler: Der Transfer wird mit Error-Status abgebrochen.

Springe zu Zeile bei Fehler und Nächste Zeile: Der Transfer wird mit dem IDN fortgesetzt, das in der in 'Nächste Zeile' angegebenen Zeile steht (die Zeilennummer ist angegeben; Zeilennummern siehe Spalte Zeile)

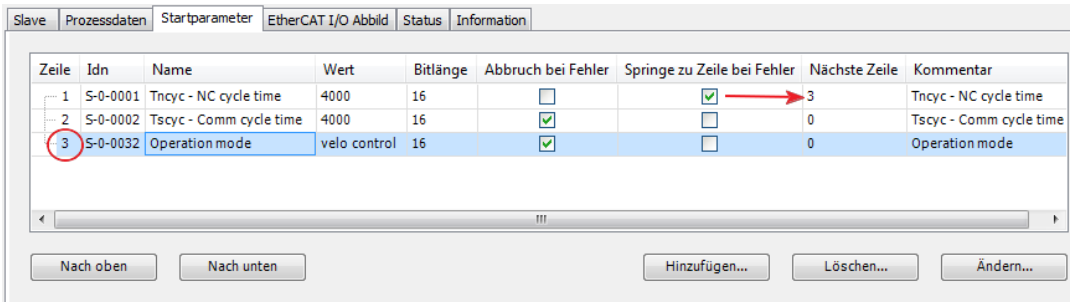


Abb. 492 Fehler-Handling konfigurieren, Beispiel

Um die Parameter eines IDNs zu verändern das bereits in der IDN-Konfigurationstabelle enthalten ist, verwenden Sie Schaltfläche Ändern... um den Dialog Wählen Sie einen Eintrag aus dem Objektverzeichnis zu erhalten, in dem Sie arbeiten können wie es oben für das Hinzufügen eines IDNs beschrieben ist.

Schaltfläche Löschen... entfernt den gerade selektierten Eintrag.

Online

Dieses Registerblatt ist nur dann im Geräte-Editor eines EtherCAT Slaves verfügbar, wenn die Experteneinstellungen für den Slave aktiviert sind und die Applikation im Gerät eingeloggt ist. Die Ansicht stellt Statusinformationen des Slaves sowie Funktionen für die Dateiübertragung zum Slave über EtherCAT zur Verfügung (FoE).

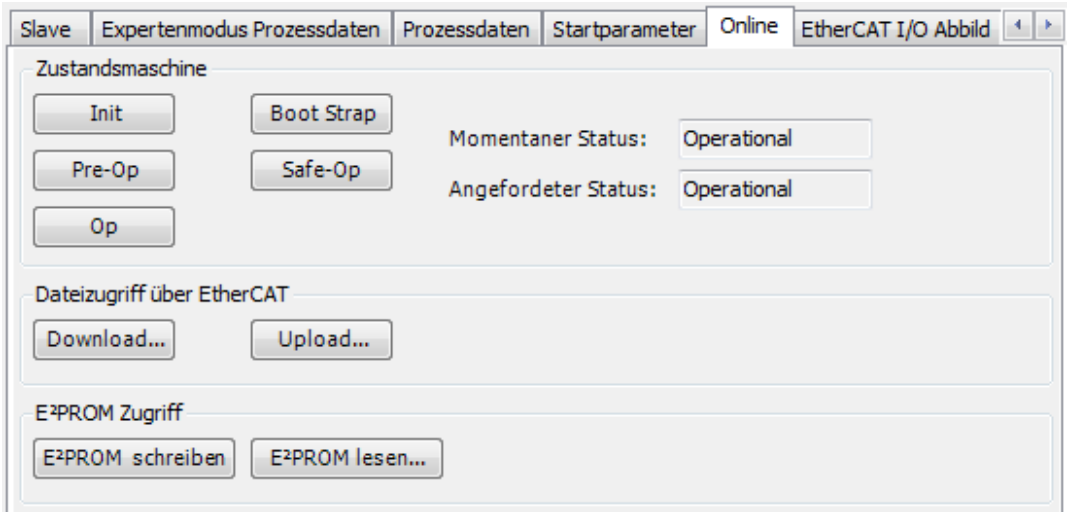


Abb. 493 Dialog für Online Dateiübertragung

Zustandsmaschine

Die Schaltflächen Init, Pre-Op, Op und Safe-Op können für Debugging-Zwecke verwendet werden.

Dateizugriff über EtherCAT

Wenn Firmware-Dateien zum oder vom Slave übertragen werden sollen, muss der Slave über die Schaltfläche Bootstrap in den 'Bootstrap Mode' geschaltet werden. Der Download und Upload von Firmware-Dateien kann mit den entsprechenden Schaltflächen ausgeführt werden. Danach öffnet ein Dialog zum Speichern bzw. Öffnen der Firmware-Datei. In diesem Dialog ist die Eingabe eines Strings sowie eines Passwortes notwendig um die Dateiübertragung auszuführen. Diese Informationen sind dem Datenblatt des Slaves zu entnehmen.

E²PROM Zugriff

Die Konfiguration des Slaves kann aus dem E²PROM gelesen oder in das E²PROM geschrieben werden. Hier öffnet analog zum Befehl für Dateizugriff über EtherCAT ein Dialog zum Öffnen bzw. Speichern der Datei.

EtherCAT Konfiguration / Slave

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Slaves zeigt alle Slave-Parameter, die in der Gerätebeschreibungsdatei definiert sind.

Wenn der Autokonfigurationsmodus im Master aktiviert ist, werden die Parameter hier automatisch gemäß den Angaben aus der Beschreibungsdatei und der Netzwerk-Topologie eingestellt. Für Standardapplikationen ist es normalerweise dann auch **nicht erforderlich, sie zu bearbeiten**.

Die Parameter werden mit **Parameter**namen, Daten**Typ**, aktuell eingestelltem **Wert**, **Standardwert**, **Einheit** und einer kurzen **Beschreibung** dargestellt.. Das Feld 'Wert' kann editiert werden, was aber nur wirksam wird, wenn der Autokonfigurationsmodus deaktiviert ist!

EtherCAT I/O Abbild / Slave

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Slaves ermöglicht, Projektvariablen auf die Ausgänge und Eingänge des Slaves abzubilden. Somit kann das Steuerungsgerät am Slave durch das Applikationsprogramm kontrolliert werden.

Wie das "Mapping" durchgeführt wird, sehen Sie bitte in der Beschreibung des allgemeinen I/O-Abbild-Dialogs im Geräteeditor.

Die automatisch angelegte Instanz des Slaves, die in der Applikation verwendet werden kann, wird unten im Dialog (IEC-Objekte) angezeigt.

Status

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Geräts zeigt Status-Informationen (z.B. "Running", "Stopped") und spezifische Diagnosemeldungen aus dem Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Information

Dieses Registerblatt im Konfigurationseditor eines EtherCAT Masters zeigt folgende allgemeine Informationen zum Gerät an, soweit sie in der Gerätebeschreibungsdatei vorliegen: Name, Hersteller, Versionsnummer, Kategorien, Bestellnummer, Beschreibung, Abbildung.

Zugriff auf EtherCAT Konfiguration aus der Applikation

Implizite Instanz des Masters

Für jeden EtherCAT Master wird implizit eine Instanz vom Typ "IoDrvEtherCAT" angelegt, sobald der Master im Gerätebaum eingefügt wurde. Der Name der Instanz entspricht dem im Baum für den Master verwendeten Namen. Die Verfügbarkeit der Instanz wird im unteren Teil (IEC Objekte) des I/O Abbild Dialogs angezeigt.

Die Master-Instanz kann in der Applikation verwendet werden, um auf den EtherCAT Bus zur Laufzeit zuzugreifen; beispielsweise um den Master neu zu starten.

Die folgenden Parameter können gemäß Syntax "<Instanzname>.<Parameter>" verwendet werden:

Eingabe-Parameter:

xRestart	Bei einer steigenden Flanke an diesem Eingang wird der Master neu gestartet. Alle Konfigurationsparameter werden neu geladen.
xStopBus	Wenn dieser Eingang TRUE ist, wird die Kommunikation gestoppt. Keine weiteren EtherCAT Telegramme werden versendet. Auf den meisten Geräten ist hiernach ein Neustart nötig, da sie in den Fehler-Status geschaltet wurden.

Ausgabe-Parameter:

xConfigFinished	Wenn dieser Ausgang TRUE ist, wurde die Übertragung aller Konfigurationsparameter fehlerfrei abgeschlossen. Die Kommunikation auf dem Bus läuft.

Beispiele für die Verwendung der Instanz:

EtherCAT_Master.xRestart := xRestart;	Neustart des Masters über Variable xRestart veranlassen
EtherCAT_Master.xStopBus := xStop;	Anhalten der Kommunikation auf dem Bus über Variable xStop veranlassen
EtherCAT_Master(); xFinish := EtherCAT_Master.xConfigFinished;	Aufruf des Masters, um Information über den Erfolg des Downloads der Konfigurationsparameter zu erhalten

Alternativ kann, sofern vom Gerät unterstützt, die von der Bibliothek IIoDrvBusControl.library zur Verfügung gestellte Schnittstelle implementiert werden, um Zugriff auf das Gerät von der IEC Applikation aus zu ermöglichen.

Implizite Instanz des Slaves

Für jeden EtherCAT Slave wird implizit eine Instanz vom Typ "ETCSlave" angelegt, sobald der Slave im Gerätebaum eingefügt wurde. Der Name der Instanz entspricht dem im Baum für den Slave verwendeten Namen. Die Verfügbarkeit der Instanz wird im unteren Teil (IEC Objekte) des I/O Abbild Dialogs angezeigt.

Die Slave-Instanz kann in der Applikation verwendet werden, um den Status des Slaves zur Laufzeit abzufragen bzw. umzuschalten.

Die folgenden Parameter können gemäß Syntax "<Instanzname>.<Parameter>" verwendet werden:

Eingabe-Parameter:

xSetOperational	Bei einer steigenden Flanke an diesem Eingang wird versucht, den Slave zurück in den ETC_SLAVE_ OPERATIONAL Modus zu schalten.

Ausgabe-Parameter:

wState	Der aktuelle Status des Slaves wird hier ausgegeben. Mögliche Werte: 0: ETC_SLAVE_BOOT 1: ETC_SLAVE_INIT 2: ETC_SLAVE_PREOPERATIONAL 4: ETC_SLAVE_SAVEOPERATIONAL 8: ETC_SLAVE_OPERATIONAL

Status ETC_SLAVE_OPERATIONAL bedeutet, dass die Konfiguration erfolgreich abgeschlossen wurde. Wenn während der Konfiguration Fehler auftreten, kann der Slave in einen früheren Status zurückfallen.

Alternativ kann, sofern vom Gerät unterstützt, die von der Bibliothek IIoDrvBusControl.library zur Verfügung gestellte Schnittstelle implementiert werden, um Zugriff auf das Gerät von der IEC Applikation aus zu ermöglichen.

Verkettete Liste zur Überprüfung aller Slaves

Grundsätzlich wird für jedem EtherCAT Master und jeden EtherCAT Slave eine Instanz eines Funktionsbausteins automatisch erzeugt. Damit kann beispielsweise der Zustand eines einzelnen Slaves überwacht werden. Im Programm muss hierzu die Instanz aufgerufen werden und dann kann über wState der Zustand ermittelt werden. Um die Programmierung zu vereinfachen, können alle Master und Slaves durch verkettete Listen ermittelt werden. Somit können durch eine einfache WHILE-Schleife sämtliche Slaves überprüft werden.

Sowohl beim Master als auch Slave gibt es die Eigenschaften **NextInstance** und **LastInstance**. Sie geben einen Zeiger auf den nächsten bzw. vorherigen Slaven zurück. Für den Master gibt es zusätzlich die Eigenschaft **FirstSlave**, die einen Zeiger auf den ersten Slave bereitstellt.

Gemäß folgendem Beispiel können alle Slaves überprüft werden:

Beispiel

Deklaration:

```
pSlave: POINTER TO ETCSlave;
```

Programm:

```
pSlave := Ethercat_Master.FirstSlave;
WHILE pSlave <> 0 DO
pSlave^();
IF pSlave^.wState = ETC_SLAVE_STATE.ETC_SLAVE_OPERATIONAL THEN
;
END_IF
pSlave := pSlave^.NextInstance;
END_WHILE
```

Es wird zunächst über Ethercat_Master.FirstSlave der erste Slave am Master geholt.

In der WHILE-Schleife wird die jeweilige Instanz aufgerufen und damit wState ermittelt. Dann kann der Status überprüft werden. Mit pSlave^.NextInstance wird der Zeiger auf den nächsten Slave geholt. Wenn die Liste zu Ende ist, ist der Zeiger NULL und die Schleife wird verlassen.

Eigenschaften des Masters:

StartConfigWithLessDevice	Hiermit kann das Startverhalten des Stacks beeinflusst werden. Der EtherCAT Stack stoppt normalerweise sofort, wenn beispielsweise im Projekt fünf Servoregler konfiguriert sind, aber tatsächlich nur 3 Regler angeschlossen sind. Wird im ersten SPS-Zyklus StartConfigWithLessDevice := TRUE gesetzt, so versucht der Stack trotzdem weiter zu starten. Es ist somit beispielsweise möglich eine universelle Konfiguration von 10 Servoreglern zu erstellen, aber die Anzahl der tatsächlich angeschlossenen variabel zu haben. Zu beachten ist, dass trotzdem die Vendor und Produkt ID jedes einzelnen Slaves überprüft wird. Sollten diese nicht übereinstimmen, dann wird der Start des Stacks gestoppt.
NumberActiveSlaves	Diese Eigenschaft gibt die tatsächliche Anzahl der angeschlossenen Slaves zurück. Ist **StartConfigWithLessDevice** auf TRUE gesetzt, kann hiermit die Anzahl der realen Geräte ermittelt werden.
EnableTaskOutputMessage	EtherCAT Nachrichten werden normalerweise von der Buszyklus-Task und zusätzlich von jeder Task gesendet, welche Ausgänge der Slaves verwendet. In der Buszyklus-Task werden alle Ausgänge geschrieben und alle Eingänge gelesen. In den weiteren Tasks werden die Ausgänge nochmals übertragen, damit diese sofort an die jeweiligen Slaves geschrieben werden. Dadurch soll die Totzeit bis zum Schreiben so weit wie möglich minimiert werden. Dies kann in Verbindung mit Distributed Clocks bei manchen Geräten zu Problemen führen: Wenn beispielsweise Servoregler nicht auf den **Syncinterrupt** synchronisiert sind, sondern den Schreibzeitpunkt für die interne Synchronisation verwenden, können in einem Zyklus mehrere Schreibzugriffe stattfinden. Wird **EnableTaskOutputMessage** auf False gesetzt, so wird ausschließlich die Buszyklustask verwendet. Zusätzliche Tasks führen nicht zu weiteren Nachrichten. Standardeinstellung: TRUE.
LastMessage:	Diese Eigenschaft gibt einen String mit der letzten Meldung des EtherCAT-Stacks zurück. Wenn der Start erfolgreich abgeschlossen wurde, dann sollte hier „All slaves done“ zu lesen sein. Der gleiche String erscheint in der Diagnosemeldung im EtherCAT Master Geräteeditor, welche auf der ersten Seite im Online-Betrieb angezeigt wird.

IODrvEtherCAT Bibliotheksbausteine für "CAN over Ethercat"

Funktionsbausteine für "CAN over EtherCAT"

Die EtherCAT-Bibliothek IODrvEtherCAT.library, die automatisch im Projekt eingebunden wird, wenn eine EtherCAT-Konfiguration unterstützt und vorgenommen wird, enthält einige Funktionsblockbaustein zum Lesen und Schreiben von Geräteparametern. Damit können die einzelnen Parameter auch zur Laufzeit geprüft und verändert werden.

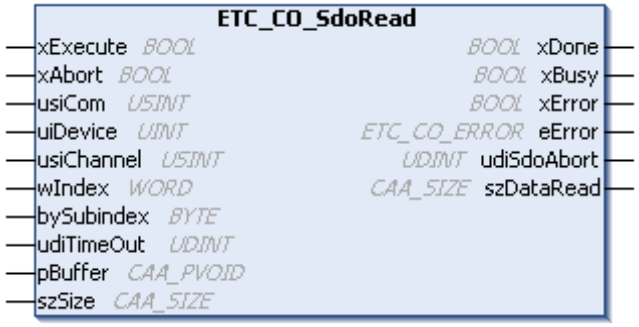
Mehrere Funktionsbausteine können gleichzeitig aktiv sein. Intern werden die einzelnen Anfragen in einer Schleife verwaltet und nacheinander abgearbeitet.

Folgende Module stehen zur Verfügung:

- ETC_CO_SdoRead (Lesen eines Parameters, der auch länger als 4 Bytes sein kann)
- ETC_CO_SdoRead4 (Lesen eines Parameters, der nicht länger als 4 Bytes ist)
- ETC_CO_SdoWrite (Schreiben eines Parameters, der auch länger als 4 Bytes sein kann)
- ETC_CO_SdoWrite4 (Schreiben eines Parameters, der nicht länger als 4 Bytes ist)

ETC_CO_SdoRead

Dieser Funktionsblock, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient dem **Lesen von EtherCAT Slave Parametern**. Im Unterschied zu ETC_CO_SdoRead4 werden auch Parameter unterstützt, die **länger als 4 Bytes** sind. Der zu lesende Parameter wird mit Index und Subindex, wie im Objektverzeichnis verwendet, angegeben.



Eingabeparameter:

xExecute	Eine steigende Flanke an diesem Eingang startet das Lesen des Slave-Parameters. Um den internen Kanal danach wieder frei zu geben, muss die Instanz mindestens einmal mit "xExecute:= FALSE" aufgerufen werden.
xAbort	Wenn dieser Parameter TRUE wird, wird der aktuelle Leseprozess abgebrochen.
usiCom	Nummer des EtherCAT Masters: Wenn nur 1 EtherCAT Master verwendet wird, ist usiCom immer "1". Wenn mehrere Master verwendet werden, bezeichnet "1" den ersten, "2" den zweiten, etc.
uiDevice	Physikalische Adresse des Slaves. Wenn der Autokonfigurationsmodus im Master deaktiviert ist, kann im Slave eine eigene Adresse gesetzt werden. Diese - beliebig gewählte - Adresse muss hier angegeben werden. Wenn der Autokonfigurationsmodus aktiviert ist, erhält der erste Slave die Adresse "1001". Die aktuelle Adresse eines Slaves kann immer im Slave Dialog des Geräte-Editors im Feld „EtherCAT Adresse“ nachgesehen werden.
usiChannel	Reserviert für künftige Erweiterungen. Zur Zeit nicht verwendet.
wIndex	Index des Parameters im Objektverzeichnis.
bySubIndex	Subindex des Parameters im Objektverzeichnis.

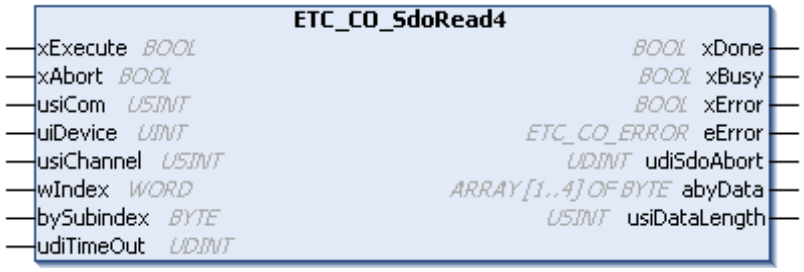
udiTimeout	Hier kann eine Überwachungszeit im Millisekunden definiert werden. Wenn das Lesen des Parameters nach Ablauf dieser Zeit noch nicht abgeschlossen ist, wird eine Fehlermeldung ausgegeben.
pBuffer	Zeiger auf einen Datenpuffer, in dem die Daten nach einer erfolgreichem Parameterübertragung abgelegt werden.
szSize	Größe des Datenpuffers (s.o.: pBuffer) in Bytes.

Ausgabeparameter:

xDone	Dieser Ausgang wird TRUE sobald das Lesen des Parameters fehlerfrei beendet ist.
xBusy	Dieser Ausgang ist TRUE solange das Lesen noch nicht beendet ist.
xError	Dieser Ausgang wird TRUE, wenn beim Lesen ein Fehler aufgetreten ist. eError gibt die Fehlerursache aus.
eError	Dieser Ausgang (Typ ETC_CO_ERROR) gibt Information über die Ursache des Fehlers, der durch xError angezeigt wurde. Z.B. "ETC_CO_TIMEOUT" im Falle einer Zeitüberschreitung.
udiSdoAbort	Wenn im Gerät ein Fehler aufgetreten ist, gibt dieser Ausgang weitere Informationen dazu.
szDataRead	Anzahl der gelesenen Bytes; maximal szSize (siehe Eingabeparameter).

etc_co_sdoread4

Dieser Funktionsblock, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient dem **Lesen von EtherCAT Slave Parametern**. Im Unterschied zu ETC_CO_SdoRead werden nur Parameter unterstützt, die **nicht länger als 4 Bytes** sind. Der zu lesende Parameter wird mit Index und Subindex, wie im Objektverzeichnis verwendet, angegeben.



Eingabeparameter:

xExecute	Eine steigende Flanke an diesem Eingang startet das Lesen des Slave-Parameters. Um den internen Kanal danach wieder frei zu geben, muss die Instanz mindestens einmal mit "xExecute:= FALSE" aufgerufen werden.
xAbort	Wenn dieser Parameter TRUE wird, wird der aktuelle Leseprozess abgebrochen.
usiCom	Nummer des EtherCAT Masters: Wenn nur 1 EtherCAT Master verwendet wird, ist usiCom immer "1". Wenn mehrere Master verwendet werden, bezeichnet "1" den ersten, "2" den zweiten, etc.
uiDevice	Physikalische Adresse des Slaves. Wenn der Autokonfigurationsmodus im Master deaktiviert ist, kann im Slave eine eigene Adresse gesetzt werden. Diese - beliebig gewählte - Adresse muss hier angegeben werden. Wenn der Autokonfigurationsmodus aktiviert ist, erhält der erste Slave die Adresse "1001". Die aktuelle Adresse eines Slaves kann immer im Slave Dialog des Geräte-Editors im Feld „EtherCAT Adresse“ nachgesehen werden.
usiChannel	Reserviert für künftige Erweiterungen. Zur Zeit nicht verwendet.
wIndex	Index des Parameters im Objektverzeichnis.
bySubIndex	Subindex des Parameters im Objektverzeichnis.
udiTimeout	Hier kann eine Überwachungszeit im Millisekunden definiert werden. Wenn das Lesen des Parameters nach Ablauf dieser Zeit noch nicht abgeschlossen ist, wird eine Fehlermeldung ausgegeben.

Ausgabeparameter:

xDone	Dieser Ausgang wird TRUE sobald das Lesen des Parameters fehlerfrei beendet ist.
xBusy	Dieser Ausgang ist TRUE solange das Lesen noch nicht beendet ist.
xError	Dieser Ausgang wird TRUE, wenn beim Lesen ein Fehler aufgetreten ist. eError gibt die Fehlerursache aus.
eError	Dieser Ausgang (Typ ETC_CO_ERROR) gibt Information über die Ursache des Fehlers, der durch xError angezeigt wurde. Z.B. "ETC_CO_TIMEOUT" im Falle einer Zeitüberschreitung.
abyData	Dies ist ein Array von 4 Bytes, in das die gelesenen Parameterdaten kopiert werden. Wenn 1 Byte gelesen wurde, wird dies in Index 1 des Arrays gespeichert. Werte von 2 bzw. 4 Bytes werden gemäß Intel Byteorder ins Array kopiert.
usiDataLength	Anzahl der gelesenen Bytes (1,2,4).

ETC_CO_SdoRreadDword

Dieser Baustein, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient wie ETC_CO_SdoRead4 dem **Lesen von EtherCAT Slave Parametern**. Jedoch werden die gelesenen Daten nicht in ein Array kopiert, sondern in ein DWORD (dwData). Eventuell notwendiges Byte Swapping wird automatisch durchgeführt. Somit kann der gelesene Wert direkt weiter verwendet werden.

ETC_CO_SdoRead_Access

Dieser Baustein, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient wie ETC_CO_SdoRead dem **Lesen von EtherCAT Slave Parametern**. Jedoch erlaubt der zusätzliche Eingang xCompleteAccess (BOOL) das Lesen des kompletten Indexes mit allen Einträgen. Dazu muss xCompleteAccess auf TRUE gesetzt sein und bySubIndex muss 0 sein.

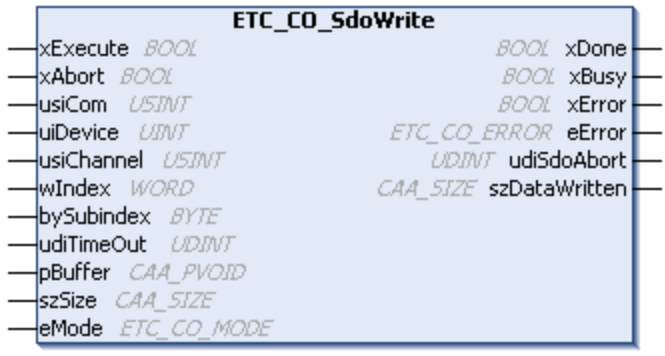
ETC_CO_SdoRead_Channel

Dieser Baustein, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient wie ETC_CO_SdoRead_Access dem **Lesen aller EtherCAT Slave Parameter**.

Es gibt jedoch einen zusätzlichen Eingang **byChannelPriority (BYTE)** mit dem Kanal und Priorität in der CoE Mailbox-Nachricht angegeben werden können. Die ersten 6 Bits geben den Kanal an, die letzten 2 Bits die Priorität.

ETC_CO_SdoWrite

Dieser Funktionsblock, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient dem **Schreiben von EtherCAT Slave Parametern**. Im Unterschied zu ETC_CO_SdoWrite4 werden auch Parameter unterstützt, die **länger als 4 Bytes** sind. Der zu schreibende Parameter wird mit Index und Subindex, wie im Objektverzeichnis verwendet, angegeben.



Eingabeparameter:

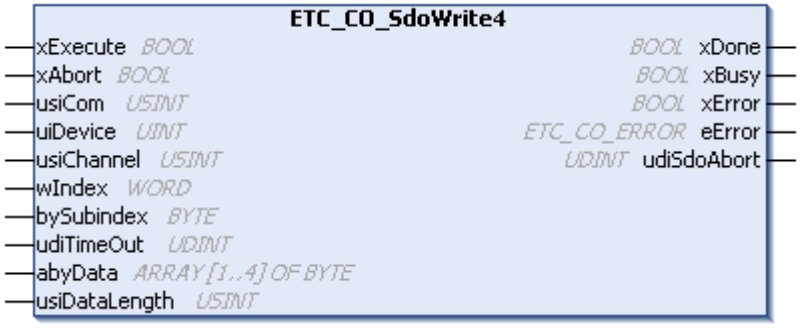
xExecute	Eine steigende Flanke an diesem Eingang startet das Schreiben des Slave-Parameters. Um den internen Kanal danach wieder frei zu geben, muss die Instanz mindestens einmal mit "xExecute:= FALSE" aufgerufen werden.
xAbort	Wenn dieser Parameter TRUE wird, wird der aktuelle Schreibprozess abgebrochen.
usiCom	Nummer des EtherCAT Masters: Wenn nur 1 EtherCAT Master verwendet wird, ist usiCom immer "1". Wenn mehrere Master verwendet werden, bezeichnet "1" den ersten, "2" den zweiten, etc.
uiDevice	Physikalische Adresse des Slaves. Wenn der Autokonfigurationsmodus im Master deaktiviert ist, kann im Slave eine eigene Adresse gesetzt werden. Diese - beliebig gewählte - Adresse muss hier angegeben werden. Wenn der Autokonfigurationsmodus aktiviert ist, erhält der erste Slave die Adresse "1001". Die aktuelle Adresse eines Slaves kann immer im Slave Dialog des Geräte-Editors im Feld „EtherCAT Adresse“ nachgesehen werden.
usiChannel	Reserviert für künftige Erweiterungen. Zur Zeit nicht verwendet.
wIndex	Index des Parameters im Objektverzeichnis.
bySubIndex	Subindex des Parameters im Objektverzeichnis.
udiTimeout	Hier kann eine Überwachungszeit im Millisekunden definiert werden. Wenn das Schreiben des Parameters nach Ablauf dieser Zeit noch nicht abgeschlossen ist, wird eine Fehlermeldung ausgegeben.
pBuffer	Zeiger auf einen Datenpuffer, der die zu schreibenden Daten enthält.
szSize	Größe der zu schreibenden Daten, in Bytes.
eMode	Dieser Einganb (Enumeration ETC_CO_MODE) definiert die Anzahl der Bytes, die geschrieben werden sollen: Mögliche Angaben: ETC_CO_AUTO, ETC_CO_EXPEDITED oder ETC_CO_SEGMENTED. Üblicherweise wird AUTO-Modus eingestellt, da dann automatisch der zur Länge passende Modus verwendet wird.

Ausgabeparameter:

xDone	Dieser Ausgang wird TRUE sobald das Schreiben des Parameters fehlerfrei beendet ist.
xBusy	Dieser Ausgang ist TRUE solange das Schreiben noch nicht beendet ist.
xError	Dieser Ausgang wird TRUE, wenn beim Schreiben ein Fehler aufgetreten ist. eError gibt die Fehlerursache aus.
eError	Dieser Ausgang (Typ ETC_CO_ERROR) gibt Information über die Ursache des Fehlers, der durch xError angezeigt wurde. Z.B. "ETC_CO_TIMEOUT" im Falle einer Zeitüberschreitung.
udiSdoAbort	Wenn im Gerät ein Fehler aufgetreten ist, gibt dieser Ausgang weitere Informationen dazu.
szDataWritten	Anzahl der geschriebenen Bytes; wird nach erfolgreich abgeschlossenem Schreiben auf szSize (siehe Eingabeparameter) zurückgesetzt .

ETC_CO_SdoWrite4

Dieser Funktionsblock, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient dem **Schreiben von EtherCAT Slave Parametern**. Im Unterschied zu ETC_CO_SdoWrite werden nur Parameter unterstützt, die **nicht länger als 4 Bytes** sind. Der zu schreibende Parameter wird mit Index und Subindex, wie im Objektverzeichnis verwendet, angegeben.



Eingabeparameter:

xExecute	Eine steigende Flanke an diesem Eingang startet das Schreiben des Slave-Parameters. Um den internen Kanal danach wieder frei zu geben, muss die Instanz mindestens einmal mit "xExecute:= FALSE" aufgerufen werden.
xAbort	Wenn dieser Parameter TRUE wird, wird der aktuelle Schreibprozess abgebrochen.
usiCom	Nummer des EtherCAT Masters: Wenn nur 1 EtherCAT Master verwendet wird, ist usiCom immer "1". Wenn mehrere Master verwendet werden, bezeichnet "1" den ersten, "2" den zweiten, etc.
uiDevice	Physikalische Adresse des Slaves. Wenn der Autokonfigurationsmodus im Master deaktiviert ist, kann im Slave eine eigene Adresse gesetzt werden. Diese - beliebig gewählte - Adresse muss hier angegeben werden. Wenn der Autokonfigurationsmodus aktiviert ist, erhält der erste Slave die Adresse "1001". Die aktuelle Adresse eines Slaves kann immer im Slave Dialog des Geräte-Editors im Feld „EtherCAT Adresse“ nachgesehen werden.
usiChannel	Reserviert für künftige Erweiterungen. Zur Zeit nicht verwendet.
wIndex	Index des Parameters im Objektverzeichnis.
bySubIndex	Subindex des Parameters im Objektverzeichnis.
udiTimeout	Hier kann eine Überwachungszeit im Millisekunden definiert werden. Wenn das Schreiben des Parameters nach Ablauf dieser Zeit noch nicht abgeschlossen ist, wird eine Fehlermeldung ausgegeben.
abyData	Dieses Array von 4 Bytes enthält die zu schreibenden Daten. Die Daten müssen gemäß Intel Byteorder abgelegt sein.
usiDataLength	Anzahl der zu schreibenden Bytes (1,2,4).

Ausgabeparameter:

xDone	Dieser Ausgang wird TRUE sobald das Schreiben des Parameters fehlerfrei beendet ist.
xBusy	Dieser Ausgang ist TRUE solange das Schreiben noch nicht beendet ist.
xError	Dieser Ausgang wird TRUE, wenn beim Schreiben ein Fehler aufgetreten ist. eError gibt die Fehlerursache aus.
eError	Dieser Ausgang (Typ ETC_CO_ERROR) gibt Information über die Ursache des Fehlers, der durch xError angezeigt wurde. Z.B. "ETC_CO_TIMEOUT" im Falle einer Zeitüberschreitung.
udiSdoAbort	Wenn im Gerät ein Fehler aufgetreten ist, gibt dieser Ausgang weitere Informationen dazu.
szDataWritten	Anzahl der geschriebenen Bytes; wird nach erfolgreich abgeschlossenem Schreiben auf szSize (siehe Eingabeparameter) zurückgesetzt.

ETC_CO_SdoRead4

Dieser Baustein, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient wie ETC_CO_SdoWrite4 dem **Schreiben von EtherCAT Slave Parametern**. Jedoch werden die zu schreibenden Daten nicht als Array übergeben, sondern in einem DWORD (dwData). Wenn Byte Swapping erforderlich ist, wird es automatisch durchgeführt. Somit kann der zu schreibende Wert direkt angegeben werden.

etc_co_sdowriteaccess

Dieser Baustein, der in der Bibliothek IODrvEtherCAT.library enthalten ist, dient wie ETC_CO_SdoWrite dem **Schreiben von EtherCAT Slave Parametern**.

Jedoch kann mit Hilfe des zusätzlichen Eingangs **xCompleteAccess (BOOL)** der komplette Index mit allen Einträgen geschrieben werden. Wenn xCompleteAccess dazu auf TRUE gesetzt wird, muss bySubIndex 0 sein. Mit Hilfe eines weiteren Eingangs, **byChannelPriority (BYTE)**, kann das entsprechende Datenelement (Kanal und Priorität) in der CoE Mailbox Nachricht angegeben werden.

14.4 Profibus DP-Konfigurationseditor

Die Funktionalität zu Erstellen einer Steuerungskonfiguration gemäß des **Profibus DP Standards** wird durch das Plug-In 'Device Editor Profibus' bereit gestellt.

Ein Profibus DP System besteht aus einem oder mehreren **Masters** und ihren **Slaves**. Jedes Slave-Modul erhält vom Master einen Satz an Parametern und stellt auf Anforderung des Masters Daten bereit. Bei der Initialisierung des Bussystems parametriert jede Master die Slaves, die ihm laut Konfiguration zugewiesen sind. In einem laufenden Bussystem, sendet der Master Daten an die Slaves und/oder fordert Daten von diesen an.

Um Profibus-Geräte als Objekt im Gerätebaum einfügen zu können, müssen **Gerätebeschreibungsdateien**, die mit der Hardware geliefert werden, im Geräte-Repository-Dialog importiert werden ("Installieren" der Geräte). Beschreibungsdateien für Profibus Module werden in verschiedenen Landessprachen bereitgestellt, und haben standardmäßig die Erweiterung **GS<Landeskennzeichner>**. Der Landeskennzeichner ist z.B. "E" für Englisch und "D" für Deutsch.

Unterhalb eines Masters können ein oder mehrere Slave-Module eingehängt werden. Unterhalb eines Profibus-Geräts können entsprechende E/A-Module eingefügt werden. Das Einfügen erfolgt mit Hilfe des Befehls Gerät anhängen bzw. Gerät einfügen oder Gerät setzen. Beachten Sie auch die Möglichkeit, ein Gerät für den Netzwerkbetrieb vorübergehend zu deaktivieren.

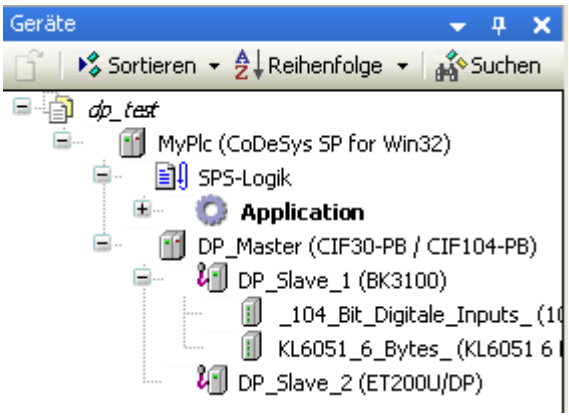


Abb. 494 Beispiel einer DP Konfiguration

Wenn ein Profibus DP Gerät im Gerätebaum selektiert ist, enthält der Geräteeditor-Dialog mindestens die Registerblätter 'DP Parameter' und 'Information'. Registerblatt '...Konfiguration' wird nur angezeigt, wenn die entsprechende Option aktiviert ist. Registerblatt '...I/O-Abbild' ist nur für Eingangs-/Ausgangs-Module verfügbar, Registerblatt 'NetX Konfiguration' nur für NetX-Feldbus-Master.

Somit können die folgenden Dialoge zur Konfiguration eines DP-Moduls bereitstehen:

- NetX Konfiguration eines NetX Masters
- DP-Parameter eines DP-Masters
- DP-Parameter eines DP-Slaves
- DP-Parameter eines DP-Modules
- Profibus DP Configuration eines DP-Masters oder DP-Slaves
- DP-Modul Konfiguration eines E/A-Moduls an einem DP-Gerät
- DP-Modul I/O-Abbild eines E/A-Moduls an einem DP-Gerät
- Status
- Information

DP-Parameter eines DP-Masters

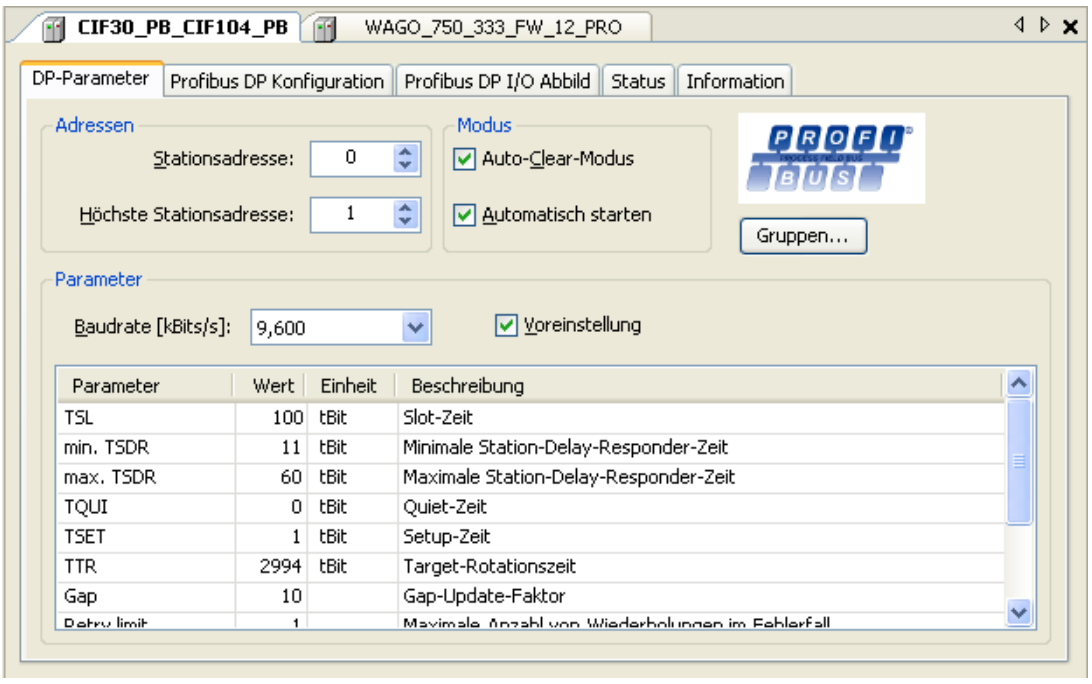


Abb. 495 DP-Parameter Dialog für einen DP-Master

Die DP-Parameter eines DP-Masters sind ein Standard-Set von Parametern, die für jeden Profibus-Master konfiguriert werden müssen. Sie sind grundsätzlich durch die Gerätebeschreibungsdatei (GS?-Datei) definiert, können jedoch im DP-Parameter-Dialog durch den Anwender noch modifiziert werden.

Adressen

Stationsadresse: Der erlaubte Bereich ist 0 bis126. Jedes neu auf dem Bus eingefügte Gerät erhält automatisch die nächst höhere Adresse (Hinweis: Adresse 126 ist die Default- DP-Slave-Adresse). Manuelles Eintragen ist möglich; Adressen werden auf doppeltes Vorkommen geprüft.

Höchste Stationsadresse: Die höchste Stationsadresse (HSA) auf dem Bus wird angezeigt. Um einen engeren GAP-Bereich (Adressbereich, in dem nach neuen aktiven Geräten gesucht wird).zu erhalten, kann hier eine niedrigere Adresse angegeben werden

Modus

Auto-Clear-Modus: Wenn diese Option aktiviert ist, werden im Fehlerfall die Ausgänge der Slaves in den sicheren Status geschaltet und der Master wechselt vom "Operate" in den "Clear"-Status.

Automatisch starten: Wenn diese Option aktiviert ist, startet der Master selbständig, ansonsten muss er manuell gestartet werden. Achtung: Es **hängt vom Treiber ab**, ob die Einstellung ausgewertet wird.

Parameter

Diese Parameter beschreiben das Zeitverhalten der Kommunikation auf dem Profibus.

Baudrate [kBits/s]: Die von der Gerätebeschreibungsdatei (GS?-Datei) definierte Auswahl steht hier zur Verfügung. Allerdings kann nur eine Baudrate eingestellt werden, die von allen Slaves unterstützt wird.

Voreinstellung: Wenn diese Option aktiviert ist, werden die Werte in der Parameter-Tabelle auf ihre Standardwerte zurückgesetzt, entsprechend der gerade eingestellten Baudrate.

Die Parameter-Tabelle zeigt für jeden **Parameter** den **Wert**, die **Einheit** und eine kurze **Beschreibung**. Die folgenden Parameter können beschrieben sein und können nach einem Mausklick auf das jeweilige Wert-Feld bearbeitet werden:

TSL [tBit]	"Slot time": Maximale Zeitspanne, die der Master nach Senden einer Anforderung auf das erste Zeichen der Antwort des Slaves wartet.
min. TSDR [tBit]	Minimale "Station Delay Responder Time": Minimale Reaktionszeit, nach der ein Teilnehmer am Bus antworten darf (min.11 TBit)
max. TSDR [tBit]	Maximale "Station Delay Responder Time": Maximale Zeitspanne, innerhalb der ein Slave antworten muss.
TQUI [tBit]	"QUIet time": Ruhezeit, die beim Umsetzen von NRZ-Signalen (Non Return to Zero) auf andere Kodierungen zu berücksichtigen ist (Umschaltzeit für Repeater)
TSET [tBit]	"Setup Time": Zeit für Setup
TTR [tBit]	"Target Time for Rotation"; Token-Soll-Umlaufzeit; projektiertes Zeitintervall, in dem ein Master den Token erhalten soll. Ergibt sich aus der Summe der Token-Halte-Zeiten aller Master am Bus.
Gap	GAP-Aktualisierungsfaktor G: Anzahl der Busumläufe, nach denen im GAP des Masters (Adressbereich von der eigenen Busadresse bis zur Adresse des nächsten aktiven Teilnehmers) nach einer weiteren, neu hinzugekommenen, aktiven Station gesucht wird.
Retry limit	Maximale Anzahl der erneuten Aufrufversuche des Masters, wenn er vom Slave keine gültige Antwort empfangen hat.
Slave interval [µs]	Zeit zwischen zwei Buszyklen, in denen ein Slave eine Anforderung des Masters bearbeiten kann (Zeitbasis 100µs). Der hier eingetragene Wert muss mit den jeweiligen Vorgaben in den GSD-Dateien der Slaves abgestimmt sein.
Poll timeout [ms]	Maximale Zeit, nach der die Antwort des Masters bei einer Master-Master-Kommunikation vom Requester (DP_Master Klasse 2) abgeholt sein muss (Zeitbasis 1ms).
Data control time [ms]	Zeit, in der der Master den zugeordneten Slaves seinen Betriebszustand mitteilt. Gleichzeitig überwacht der Master, ob mit den Slaves innerhalb dieser Zeit jeweils mindestens ein Nutzdatenaustausch stattgefunden hat und aktualisiert die Data_Transfer_List.

tBit: Zeiteinheit für die Übertragung eines Bits auf dem PROFIBUS; Reziproker Wert der Übertragungsrate; z.B. 1 tBit bei 12MBaud=1/12.000.000 Bit/sec=83ns

Gruppen: Diese Schaltfläche öffnet den Dialog zum Konfigurieren der globalen Gruppeneigenschaften.

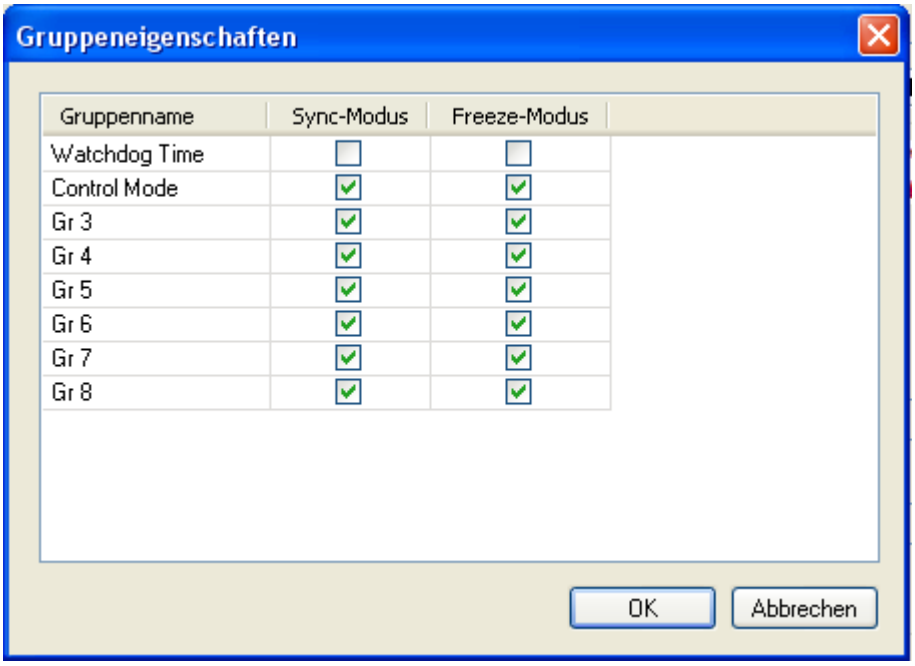


Abb. 496 Gruppeneigenschaften Dialog

Bis zu acht Gruppen können definiert werden, mit verschiedenen Kombinationen von Freeze-Modus und/oder Sync-Modus Eigenschaften, das heißt der Aktivierung des Freeze und/oder Sync Modus für den Datenaustausch. Die Slaves können dann einer oder mehreren Gruppen zugeordnet werden (siehe DP Parameter eines DP-Slaves). Dies macht es dem Master möglich, den Datenaustausch zu synchronisieren. Durch ein globales Kontrollkommando (Freeze, Sync) kann er alle Slaves einer Gruppe gleichzeitig instruieren.

Mit einem **Freeze** Kommando befiehlt der Master einem Slave oder einer Gruppe, alle Eingangswerte im momentanen Zustand einzufrieren und diese Daten in der nachfolgenden Datenaustauschaktion zu übertragen. Mit einem **Sync** Kommando werden die Slaves angewiesen, alle Daten, die sie in der nachfolgenden Datenaustauschaktion vom Master erhalten werden, synchron weiterzureichen.

Die **Gruppennamen** können verändert werden. Zu diesem Zweck muss der Name in der Spalte **Gruppe** selektiert und die <Leertaste> gedrückt werden, um die Zeichenfolge editierbar zu bekommen. Um die Freeze und Sync Option für eine Gruppe zu aktivieren, muss ein Mausklick auf die Checkbox in der jeweiligen Spalte ausgeführt werden. Ein "x" in dieser Box bedeutet, dass die Eigenschaft aktiviert ist.

DP-Parameter eines DP-Slaves

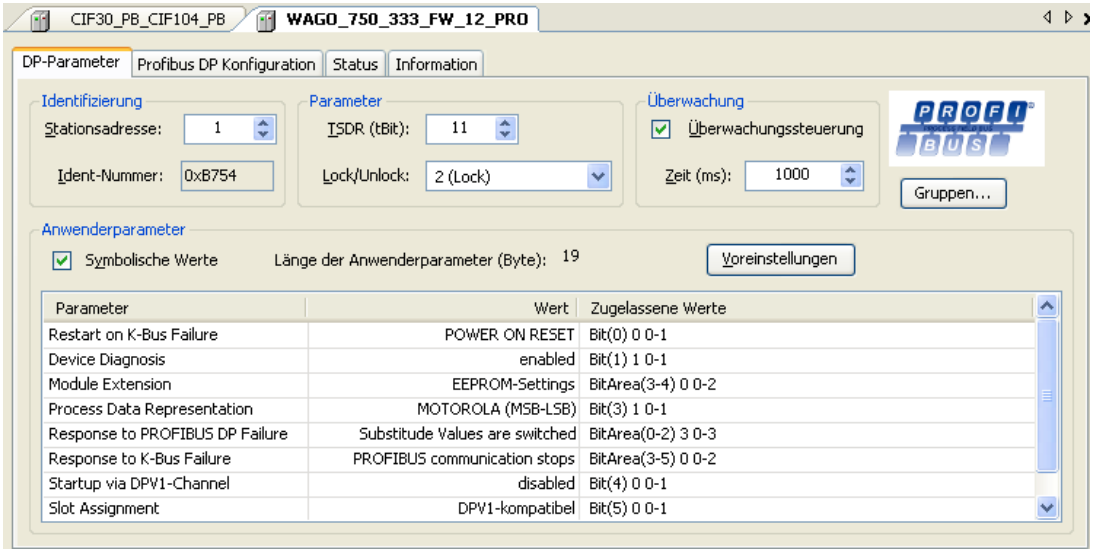


Abb. 497 DP-Parameter Dialog für einen DP-Slave

Die DP-Parameter eines DP-Slaves sind im Unterschied zu denen des Masters nicht ein Standard-Set, sondern werden in der Gerätebeschreibung individuell für jedes Gerät definiert. Sie können im DP-Parameter-Dialog der Profibus-Gerätekonfiguration durch den Anwender modifiziert werden.

Identifizierung

Stationsadresse: siehe DP Master

Ident-Nummer: Eindeutige Nummer, die vom PNO diesem Gerätetyp zugewiesen wurde. Dies erlaubt eine eineindeutige Referenz zwischen DP-Slave und zugehöriger GS?-Datei.

Parameter

TSDR (tBit): Time Station Delay Responder. Reaktionszeit, nach der der Slave frühestens an den Master antworten darf (min. 11 TBit)

* TBit: Zeiteinheit für die Übertragung eines Bits über PROFIBUS; Kehrwert der Übertragungsrate; z.B. 1 TBit bei 12MBaud=1/12.000.000 Bit/sek=83ns

Lock/Unlock: Slave wird für andere Master gesperrt oder freigegeben:

0 (TSDR unlock):	min.TSDR und slave-spezifische Parameter dürfen überschrieben werden
1 (Will be unlocked):	Slave für andere Master freigegeben
2 (Lock):	Slave für andere Master gesperrt, alle Parameter werden übernommen
3 (Unlock):	Slave für andere Master erneut freigegeben

Überwachung:

Überwachungssteuerung: Wenn diese Option aktiviert ist, gilt die eingetragene Überwachungszeit (Watchdog, (Ansprechüberwachung, Basis 10 ms). Wird der Slave innerhalb dieser Zeit nicht vom Master angesprochen, geht er in den Initialisierungszustand zurück.

Zeit (ms): Überwachungszeit, relevant für den Fall, dass die Option 'Überwachungssteuerung' aktiv ist.

Anwenderparameter

"Anwenderparameter" sind, zusätzlich zu den Basis-DP-Parametern (s.o.), individuelle Parameter eines DP-Slaves, die - wenn in der Gerätebeschreibungsdatei (GS?-Datei) definiert - hier angezeigt werden.

Die Parametertabelle zeigt für jeden **Parameter** den Parameternamen, den echten bzw. den symbolischen **Wert** (s.o., Option 'Symbolische Werte') und die **Zugelassenen Werte** , die auch in der Gerätebeschreibungsdatei definiert sind. Die Parameterwerte können nach einem Mausklick auf das entsprechende Feld in Spalte 'Wert' editiert werden

Symbolische Werte: Wenn in der Gerätebeschreibungsdatei (GS?-Datei) auch symbolische Namen für die Parameter angegeben sind, kann diese Option hier aktiviert werden, um diese symbolischen anstelle der realen Werte in der Spalte 'Wert' angezeigt zu bekommen.

Länge der Anwenderparameter (Byte): Aufsummierte Länge der Anwenderparameter, die über die Gerätebeschreibungsdatei definiert sind.

Voreinstellungen: Über diese Schaltfläche werden die in der Tabelle angezeigten Werte auf die Standard-Einstellung zurückgesetzt.

Gruppen

Diese Schaltfläche öffnet den Dialog Gruppeneigenschaften. Hier kann der Slave einer oder mehreren von acht möglichen Gruppen zugeordnet werden, die dazu dienen den Datenaustausch zwischen Master und Slaves zu synchronisieren.

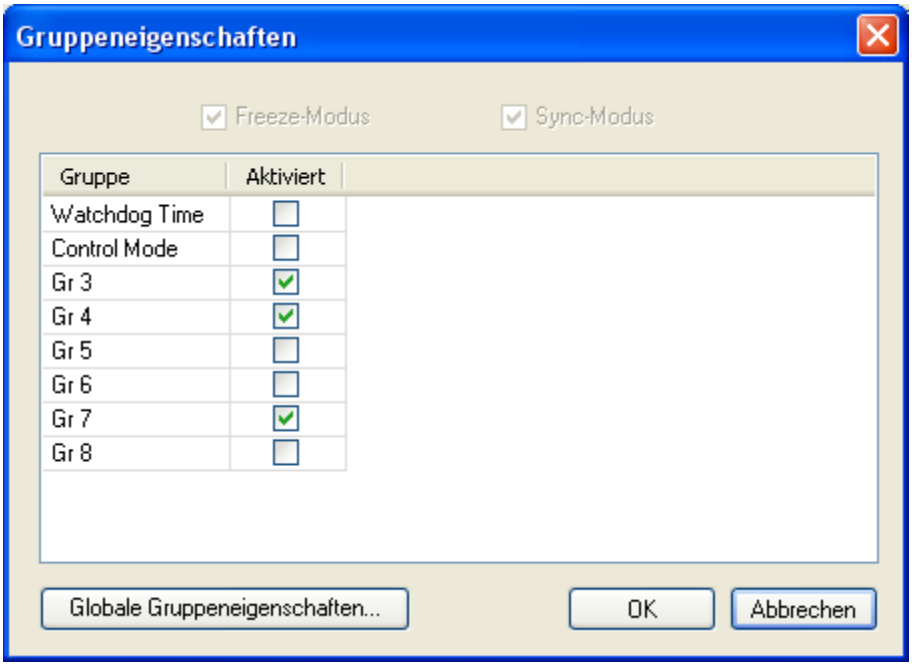


Abb. 498 Gruppeneigenschaften Dialog

Die globalen Gruppeneigenschaften (**Sync. Mode** und/oder **Freeze Mode**) werden in der DP Master-KOnfiguration definiert (siehe DP Parameter eines DP-Masters, Gruppeneigenschaften). Dieser Konfigurationsdialog kann auch von hier über die Schaltfläche **Globale Gruppeneigenschaften...** geöffnet werden.

Ein Slave-Gerät kann nur solchen Gruppen zugeordnet werden, deren Eigenschaften er unterstützt. Die betreffemdem Eigenschaften jedes Slaves sind oberhalb der Tabelle angezeigt, die unterstützten Modi sind mit einem Haken versehen.

Um einen Slave einer Gruppe zuzuordnen, klicken Sie in die Spalte **Aktiviert** um einen Haken zu setzen. Um einen Slave aus einer Gruppe wieder herauszunehmen, kann der Haken auf gleiche Weise wieder entfernt werden.

DP-Parameter eines DP-Moduls

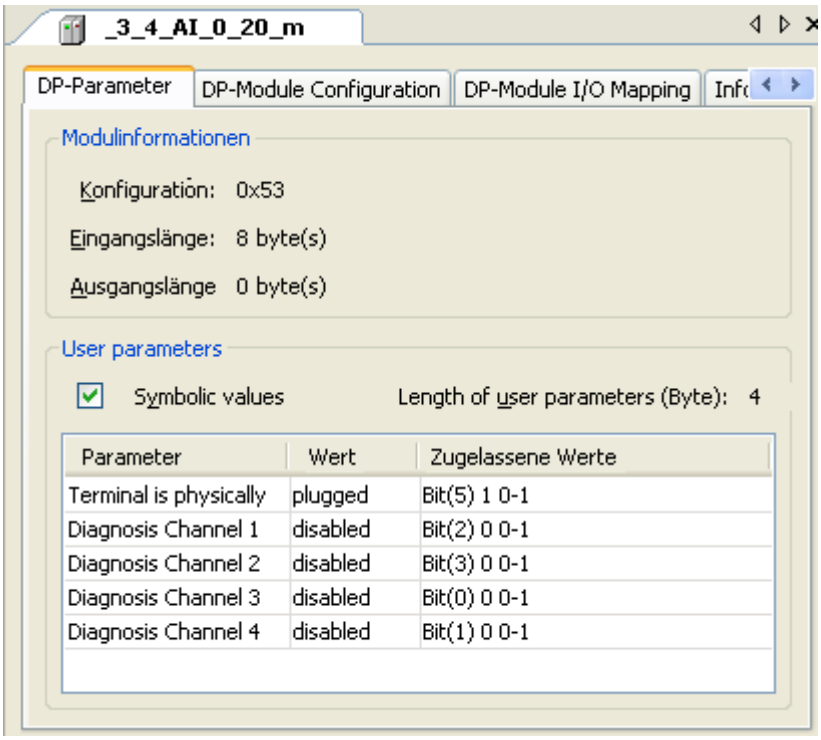


Abb. 499 DP-Parameter Dialog eines DP-Moduls

Der DP-Parameter-Dialog eines Ein-/Ausgangsmoduls in einer Profibus-Gerätekonfiguration beschreibt in Sektion Modulinformationen die Einstellungen für die **Konfiguration** (Modulbeschreibung gemäß PROFIBUS-Standard) und die Länge der Eingangs- und Ausgangsdaten (**Eingangslänge, Ausgangslänge)** in Bytes, wie sie in der Gerätebeschreibungsdatei (GS?-Datei) definiert sind.

Außerdem werden - wenn in der Gerätebeschreibung enthalten - die gerätespezifischen **Anwenderparameter** (User parameters) in einer Tabelle angezeigt, wie auch für die DP-Parameter eines DP-Slaves beschrieben. Die Option **Symbolische Werte** (Symbolic values) kann aktiviert werden, um in der Spalte 'Wert' anstelle der reellen die symbolischen Werte der Parameter anzuzeigen. Die **Länge der Anwenderparameter** (Length of user parameters) gibt die Gesamtlänge (Bytes) aller Anwenderparameter aus der Gerätebeschreibung an. Nach einem Mausklick auf das jeweilige Feld in der Spalte 'Wert' kann ein Parameterwert editiert werden.

Profibus DP Konfiguration

Dieser Dialog ist nur dann im Geräteeditor für einen Profibus Master bzw. Slave verfügbar, wenn die Option 'Generische Konfigurationsdialoge anzeigen' aktiviert ist. Er entspricht dem Dialog '..Konfiguration' für andere Gerätemodule.

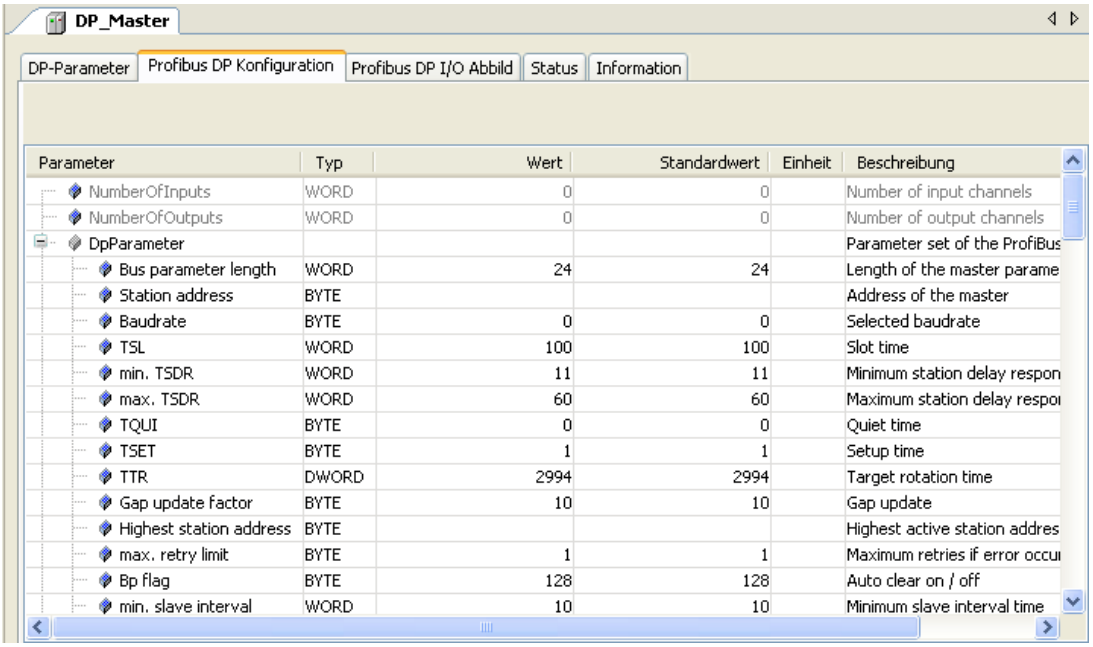


Abb. 500 Dialog 'Profibus DP Konfiguration'

Die gerätespezifischen Master- oder Slave-Parameter, die vom Hersteller in einer Gerätebeschreibungsdatei (extra Datei, nicht in der GS?-Datei!) beschrieben sind, werden hier in einer Tabelle angezeigt und können, wenn in der Gerätebeschreibung erlaubt, editiert werden. Nicht-editierbare Parameter werden grau dargestellt.

DP-Modul Konfiguration

Dieser Dialog ist nur dann im Geräteeditor für ein Profibus-Modul verfügbar, wenn Option 'Generische Konfigurationsdialoge anzeigen' aktiviert ist.. Er entspricht dem Dialog '..Konfiguration' für andere Gerätemodule.

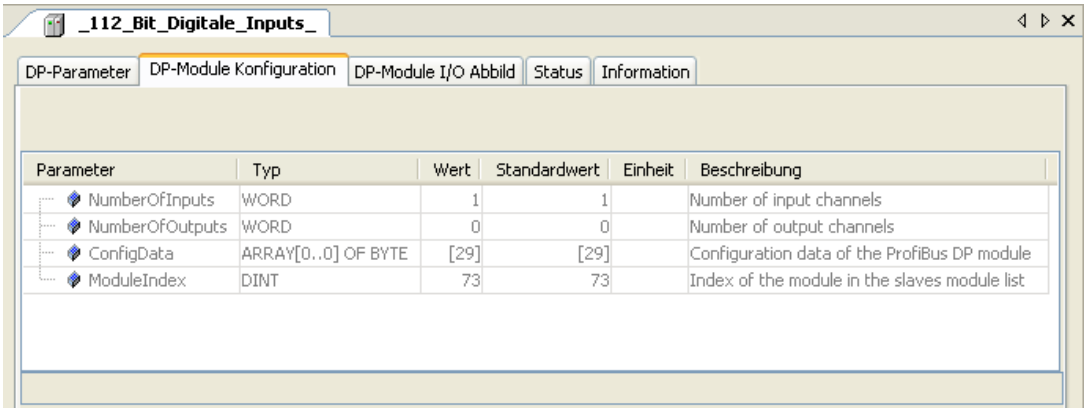


Abb. 501 DP-Modul Konfigurationsdialog

DP-Modul I/O Abbild

Dieser Dialog im Geräteeditor für ein Profibus-Modul dient dazu, die Eingänge und Ausgänge des Moduls auf Projektvariablen abzubilden. Die Handhabung entspricht derjenigen des auch für andere Geräte verwendbaren I/O-Mapping-Dialogs.

Profibus DP I/O Abbild Master

Dieser Dialog im Geräteeditor für ein Profibus-Modul dient dazu, die Eingänge und Ausgänge eines Profibus-Geräts auf Projektvariablen abzubilden. Die Handhabung entspricht derjenigen des auch für andere Geräte verwendbaren I/O-Mapping-Dialogs.

NetX Konfiguration

Bei einem NetX Feldbus ist dieser Dialog als im Geräte-Editor des Masters verfügbar, um den NetX Chip (Steckplatz der Karte) und den Kommunikationskanal auf diesem Chip auszuwählen. Der Name der Einstellung im Dialog und die möglichen Einstellungen, die in den Auswahllisten bereitstehen, werden durch die Gerätebeschreibung definiert.

Slot: Zu verwendender Steckplatz. Im Fall von PCI-Karten mit NetX Chip entsprechen die Steckplatznummern (Slots) üblicherweise den Kartennummern.

NetX Com Channel: Zu verwendender Kommunikationskanal auf der gewählten Karte. Eine NetX-Karte kann bis zu vier Kommunikationskanäle zu verschiedenen Feldbussen haben.

Bei Profibus **NetX** Konfigurationen kann mit Hilfe der Methode **GetLiveList** (verfügbar in der automatisch eingebundenen Bibliothek IoDrvCifXProfibus.library) überprüft werden, welche **Busteilnehmer** in der angebundenen Hardware vorhanden sind. Auf eine steigende Flanke des Eingangs xEnable fordert die Methode eine aktuelle Liste aller am Profibus angeschlossener Teilnehmer an. Die Operation ist abgeschlossen, wenn der Ausgang wOpState den Wert HIL_DONE oder HIL_ERROR annimmt. Die Methode kann folgende Werte zurückliefern: 00 = Station nicht verfügbar, 01 = inaktive Station, 03 = aktive Station

Status

Dieser Dialog wird allgemein vom Geräte-Editor zur Anzeige von Status und Diagnoseinformationen für das Gerät bereitgestellt.

Information

Dieser Dialog wird allgemein vom Geräte-Editor zur Anzeige einiger genereller Informationen zum Gerät bereitgestellt.

Schnittstellen zu Profibus DP Master und Slave

Die unten aufgelisteten Bibliotheken stellen Schnittstellen zu ProfibusDP-Geräten zur Verfügung. Ein Treiber kann eines der Interfaces implementieren und so den Anwender mit der zugehörigen Funktionalität ausstatten.

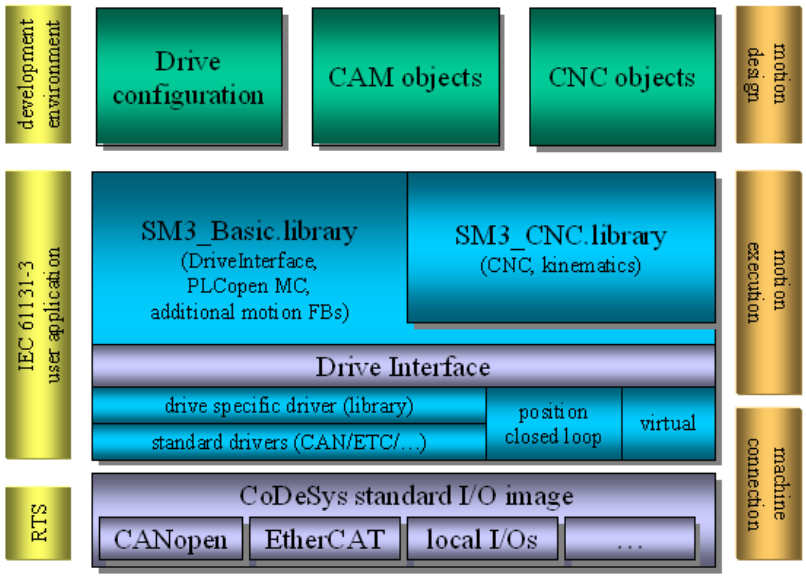
- IIoDrvBusControl.Library
- IIoDrvDPV1C1.Library
- IIoDrvDPV1C2.Library
- IIoDrvProfibus.Library

14.5 SoftMotion-Geräteeditor

14.5.1 SoftMotion Konzept und Übersicht über Komponenten

SoftMotion erlaubt Ihnen SPS gesteuerte Bewegungen effizient und ohne Erfordernis detaillierter Fachkenntnisse zu realisieren: einfache einachsige Bewegungen oder Kurvenscheiben bis hin zu komplexen Bewegungen in mehreren Dimensionen. Insbesondere Applikationen, die nicht allein die Bewegungsfunktionalität fokusieren, sondern auch Ablauf- und Prozesskontrolle oder Hilfsfunktionen, fallen in den Anwendungsbereich von SoftMotion im Rahmen der »PLC Designer« Entwicklungsumgebung. Während die Steuerlogik vollständig von der SPS übernommen wird, wird die reine Bewegungsinformation von Bibliotheksbausteinen abgearbeitet.

»PLC Designer« SoftMotion ist ein Softwarepaket, das als Entwicklungs- und Laufzeitumgebung für Motion-Steuerungen dient. Zusätzlich zu den Standardeigenschaften von »PLC Designer« und »PLC Designer« SP sind die folgenden steuerungsspezifischen Software-Teile von SoftMotion enthalten:



- Die **Drive Configuration** stellt einen Editor zur Verfügung mit dessen Hilfe Struktur und Konfiguration der Antriebs-Hardware in die »PLC Designer« Benutzer-Schnittstelle importiert werden können. Dabei erzeugen die Funktionen der Drive-Interface-Bibliotheken IEC-Funktionsblöcke als abstraktes Abbild der Antriebe. Das Drive Interface kommuniziert automatisch, d.h. ohne zusätzlichen Aufwand auf Seiten des IEC-Programmierers, mit den Antrieben und sorgt für die Aktualität der Datenstrukturen des Antriebs wie auch für die Übermittlung von aktualisierten Daten.
  Zur Kontrolle der Antriebe kann das IEC Programm die abstrakten Datenstrukturen entweder über die Standardmodule der SoftMotion Bibliotheken (SM3_Basic.library, SM3_CNC.library) ansprechen oder über vom IEC-Programmierer für diesen Zweck geschriebene Module. Dabei werden die Zielvorgaben zyklisch geschrieben; d.h., dass in jedem IEC-Task-Zyklus Sollwerte (für Position, Geschwindigkeit, Beschleunigung, etc.) einmalig berechnet und vom Drive Interface auf die Antriebe übertragen werden.
  Die Möglichkeit Antrieben einen Auftrag zu erteilen (z.B. nur eine Endposition vorzugeben, die vom Antrieb eigenständig angefahren wird und deren Erreichen er anschließend meldet) ist nicht vorgesehen. Der Grund ist, dass die zentrale Steuerung während der Abarbeitung des Auftrags keinen Einfluss auf die Antriebe hat. So wird beispielsweise eine koordinierte Bewegung mehrerer Achsen ausgeschlossen.
- Der **CAM-Editor** ist ein unabhängiges Plug-in von »PLC Designer«, das in die Programmier-Schnittstelle eingebunden werden muss. Kurvenscheiben werden zur Kontrolle mehrachsiger Antriebe verwendet. Innerhalb des Editors kann eine Kurvenscheibe graphisch oder mit Hilfe von Tabellen implementiert werden. Daraus wird eine globale Instanz einer Datenstruktur erzeugt, welche die Kurvenscheibe beschreibt. Diese wird an die Applikation übergeben, wo die geeigneten POUs auf sie zugreifen können.
- Der CNC-Editor ist ein unabhängiges Plug-in von »PLC Designer«. Er erlaubt die Programmierung mehrdimensionaler Bewegungen, die anschließend von dem mit der Kommunikation mit der Antriebs-Hardware beauftragten Drive Interface übertragen und gesteuert werden. Der Editor arbeitet angelehnt an die CNC Sprache DIN66025 synchron als graphischer oder Text-Editor. Außerdem kann die Tabellenansicht an Stelle des Text-Editors verwendet werden, die über einen ähnlichen Funktionsumfang verfügt. Grundsätzlich können Bewegungen in bis zu 9 Dimensionen realisiert werden, wobei nur 3 Dimensionen nicht linear interpoliert werden. In diesen drei Dimensionen können also Kreise, Parabeln, Ellipsen und Splines programmiert werden. Für jeden erstellten Pfad erzeugt »PLC Designer« automatisch eine Datenstruktur, die dem IEC Programm zur Verfügung gestellt wird.

- Die Bibliothek **SM3_Basic.library** ist ein Basisbaustein, welcher in alle SoftMotion Applikationen eingebunden werden muss. Sie besteht aus den folgenden Komponenten:
  - PLCopen Funktionsblöcke gemäß dem PLCopen Standard, welche eine einfache Realisierung der Steuerung einer einachsigen Bewegung erlauben wie auch die synchronisierte Bewegung zweier Achsen. Neben Bibliothekselementen zur Überprüfung des Status, zum Parametrieren und zur generellen Handhabung gibt es Funktionsblöcke, welche die Achsen unter Berücksichtigung fest vorgeschriebener Werte für Geschwindigkeit oder Beschleunigung ansteuern. Sollen zwei Achsen synchronisiert werden, so dient eine der Achsen als Master und steuert die zweite (Slave) Achse nach einer gewissen Vorschrift. Diese Vorschrift kann beispielsweise eine Kurvenscheibe sein, welche zur Anbindung des Slave an den Master Gebrauch macht von geeigneten POUs. Des weiteren gibt es Funktionsblöcke für elektronische Getriebe oder Phasenverschiebung.
  - Einige nicht standardmäßige FBs zur Bewegungskontrolle
  - Grundlegende FBs des DriveInterface (AXIS_REF, AXIS_REF_VIRTUAL, AXIS_REF_MAPPING)
  - Hilfsfunktionen: z.B. zur Handhabung von Dateien oder für Fehlermeldungen
  - Funktionsblöcke für die Implementierung spezieller Antriebstreiber: die FBs des Basistreibers können zur Kommunikation mit einem bestimmten Antriebstyp um spezifische Treiber erweitert werden.
- Die Bibliothek **SM3_CNC.library** basiert auf der Bibliothek SM3_Basic.library. Neben einigen Funktionsblöcken für kinematische Transformationen stellt sie alle POUs zur Verfügung, die zur Erzeugung, zum Ausführen und Visualisieren von CNC Bewegungen benötigt werden.
- Das **Drive Interface** ist der Teil der SM3_Basic.library, der Sorge trägt für die Kommunikation zwischen IEC Programm und Antrieben. Er ermöglicht es die FBs des Basis Treibers um spezielle Treiber zu erweitern. Diese speziellen Treiber-FBs (namens AXIS_REF_*) werden typischerweise in einer eigenen Bibliothek zusammengefasst, auf die in der Gerätebeschreibungsdatei des zugehörigen Geräts hingewiesen wird. Zusätzlich beinhaltet das Drive Interface Bus-spezifische Bibliotheken zur Handhabung des I/O Abbilds.

Hinweis: Beachten Sie die Beschreibung einiger Beispielapplikationen.

Portabilität

Mit Ausnahme spezieller Treiber, die direkt Hardware-Komponenten bedienen, sind alle SoftMotion Laufzeit-Komponenten in IEC 61131-3 programmiert, wodurch die Plattformunabhängigkeit maximiert wird.

Hinweis: Die Registerkarten Kommunikationseinstellungen und Applikationen betreffen die Kommunikation mit der Steuerung und werden vom Plug-In Device Communication Editor bereit gestellt.

14.5.2 Das SoftMotion Drive Interface

Das SoftMotion Drive Interface ist eine standardisierte Schnittstelle, die das Einbinden, Konfigurieren und Adressieren eines abstrakten Abbilds der Antriebs-Hardware innerhalb des IEC Programms erlaubt. Das Abbilden unterschiedlicher Hardware auf eine allgemeine Schnittstelle erlaubt nicht nur ein einfaches Auswechseln von Antrieben und die Wiederverwendbarkeit von IEC Programmen, sondern erleichtert auch die Kommunikation zwischen der Steuerung und den Antrieben bzw. den Antrieben untereinander. Es ist die Schnittstelle, die den Antrieb an sein Abbild in der Applikation (AXIS_REF) koppelt und verantwortlich ist für die Aktualisierung und Übertragung der notwendigen Bewegungsdaten an die Antriebssteuerung.

Das Drive Interface besteht aus den folgenden Komponenten:

- die Gerätebeschreibung der SoftMotion Geräte zu ihrer Darstellung im Gerätebaum
- in der Gerätebeschreibung referenzierte Bibliotheken, welche die grundlegenden Funktionsblöcke AXIS_REF FB erweitern oder überladen entsprechend den Anforderungen der spezifischen Antriebstypen
- Bibliotheken, die FBs enthalten für ein azyklisches Lesen/Schreiben von Daten und um Standardfunktionen für den Feldbus Treiber zu wrappen

Das Einbinden der Basis-Bibliotheken in den Bibliotheksverwalter erfolgt automatisch bei Verwendung einer **SoftMotion PLC**, welche auch eine **SoftMotion General Drive Pool** mitbringt. Die Antriebsgeräte können nun auf zwei unterschiedliche Arten hinzugefügt werden:

- **Konfiguration als "freie" Antriebsgeräte**
  Diese Anriebsgeräte sind nicht fest verbunden mit einem anderen Gerät im Gerätebaum, sondern im SoftMotion General Drive Pool eingehängt, der Bestandteil jeder SoftMotion Steuerung ist.
  Virtuelle Antriebe beispielsweise sind freie Antriebsgeräte, da sie nicht fest mit einem Feldbus-Gerät verbunden sind; auch positionsgesteuerte Antriebe können freie Geräte sein, da es keine feste Vorgabe gibt bei der Wahl eines Feldbusgeräts für ihre Ein-und Ausgänge.

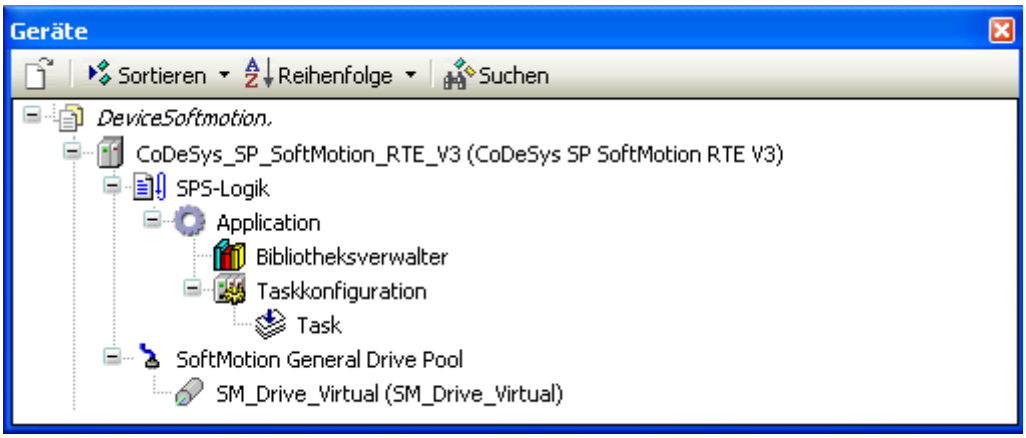


Abb. 502 Beispiel Konfiguration mit freien Antriebsgeräten

- **Konfiguration als Gerät-gekoppelte Antriebsgeräte**
  Dies Antriebsgeräte stehen in Wechselbeziehung mit einem anderen Gerät, das den Antrieb innerhalb der Feldbus-Topologie repräsentiert. Die Piktogramme dieser Antrieb sind um kleine Zeichen erweitert, welche den zugehörigen Feldbus-Typ andeuten sollen. Beispielsweise ist jeder Standard Servo Antrieb durch einen Feldbus mit der Steuerung verbunden.

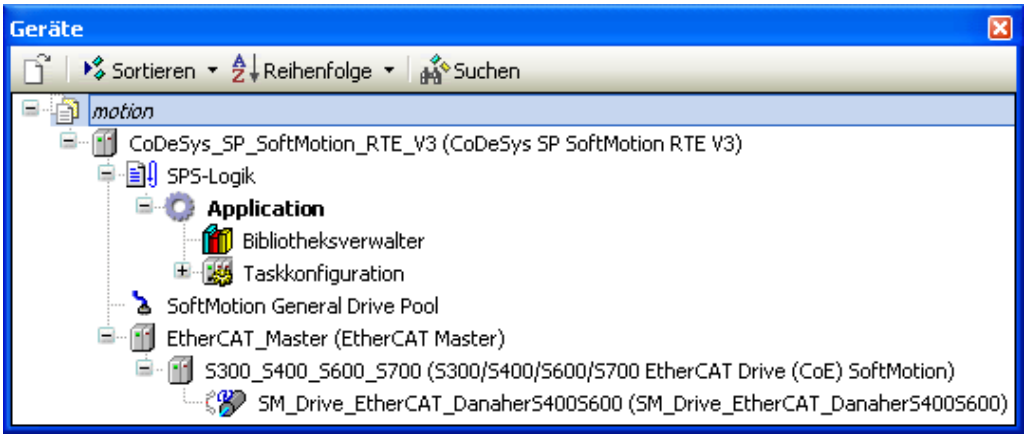


Abb. 503 Beispiel Konfiguration mit Gerät-gekoppelten Antriebsgeräten

14.5.3 Konfiguration mit Hilfe des SoftMotion Geräte-Editors

Abhängig vom Typ des Geräts enthält der SoftMotion Geräte-Editor eine unterschiedliche Auswahl an Registerkarten zur Konfiguration. Sie finden hier diejenigen Registerkarten beschrieben, die Teil der Grundausstattung des SoftMotion Editors sind:

- SoftMotion Antrieb: Basisparameter
- SoftMotion Antrieb: Skalieren/Mapping
- Status
- Information

SoftMotion Antrieb: Basisparameter

Der erste Reiter des SoftMotion Geräte-Editors enthält mehrere Eingabefelder zur Konfiguration der grundlegenden Einstellungen des eingefügten Geräts.

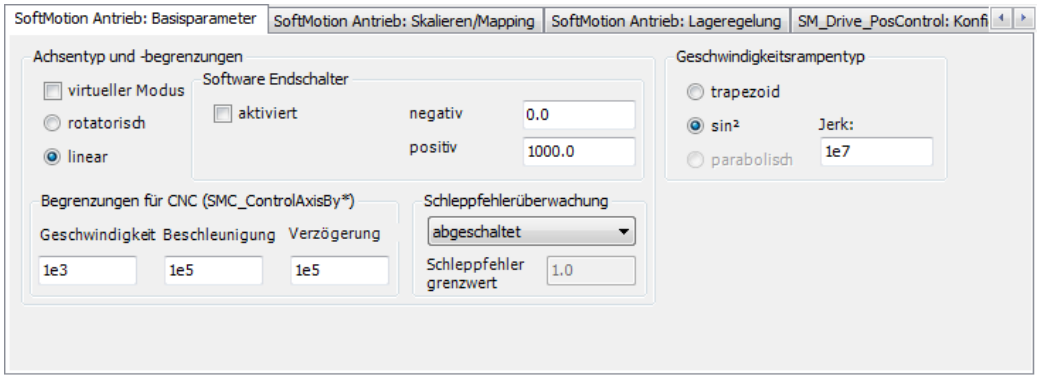


Abb. 504 SoftMotion Geräte-Editor - SoftMotion Antrieb: Basisparameter

Achsentyp und -begrenzungen

virtueller Modus: Bei Aktivierung des Kontrollkästchens "virtueller Modus" wird der Antrieb durch eine Simulation ähnlich dem virtuelle Antriebsgerät ersetzt. Entsprechend wird dieses Kontrollkästchen für virtuelle Antreibe immer aktiviert sein. Im Fall von Gerät-gekoppelten Geräten hat die Aktivierung des "virtuellen Modus" keinerlei EInfluss auf die Feldbus Geräte, die wie gewohnt arbeiten werden, jedoch ohne Nachrichten vom wirklichen Gerät zu erhalten oder zu diesem zu senden.

Sie haben die Auswahl zwischen zwei Achstypen: **linear** (finit) oder **rotatorisch** (modulo). Abhängig von Ihrer Wahl wird der rechts stehende Eingabesektor beschriftet sein mit

- **Modulo-Einstellungen** (obligatorische Einstellungen für rotatorische Antriebe): hier muss der Modulo Wert angegeben werden.
- **Software Endschalter** (mögliche Einstellungen für lineare Antriebe): aktivieren Sie das Kontrollkästchen, falls Sie die Positionswerte durch eine untere (**negative**) oder obere (**positive**) Grenze beschränken wollen.

Begrenzungen für CNC (SMC_ControlAxisBy*)

In diesem Sektor werden Grenzwerte vorgegeben (für Geschwindigkeit, Beschleunigung oder Abbremsen), welche von den Bibliotheksbausteinen SMC_ControlAxisBy* zum Auffinden von Sprüngen verwendet werden (z.B. SMC_ControlAxisByPos).

Schleppfehlerüberwachung:

Hier wird die Reaktion des Systems auf einen erkannten Schleppfehler festgelegt. Mögliche Reaktionen sind: **abgeschaltet**, **Antrieb ausschalten**, **Quickstop durchführen** und **eingeschaltet bleiben**. Ein Schleppfehler wird detektiert, wenn der Unterschied zwischen Sollposition und kompensierter Istposition den Grenzwert des Schleppfehlers überschreitet.

Schleppfehlergrenzwert: hier setzt man den Grenzwert des Schleppfehlers.

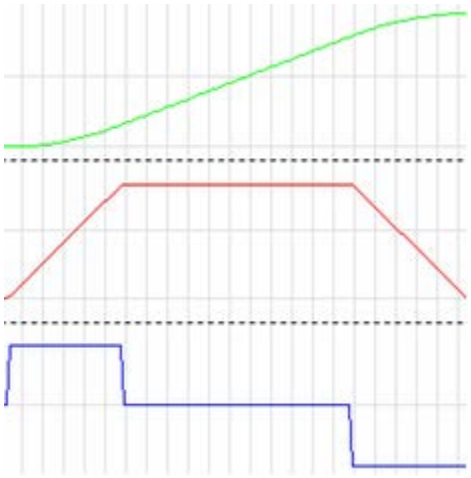
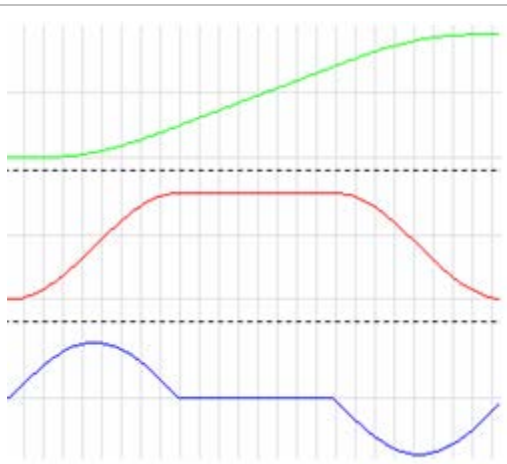
Hinweis: Es handelt sich hier um die Schleppfehlerüberwachung in der Steuerung. Eine davon unabhängige Überwachung kann zusätzlich im Antrieb existieren, die nicht über diesen Dialog konfiguriert wird.

Geschwindigkeitsrampentyp

Der Geschwindigkeits-Rampentyp legt das Geschwindigkeitsprofil fest für bewegungserzeugende Einachsbausteine und für Master/Slave Module. Die zugehörigen Bibliotheken sehen folgende Alternativen vor: **trapezoid** führt zu einem trapezförmigen Geschwindigkeitsprofil (mit konstanter Beschleunigung in jedem Abschnitt), **sin²** führt zu einem durch die Funktion sin² beschreibbaren Geschwindigkeitsprofil (mit stetigem Beschleuingungsverlauf).

Für letztes muss zusätzlich der **Jerk** festgelegt werden.

Die folgenden Bilder demonstrieren den Effekt der unterschiedlichen Rampentypen: die Position ist in grün gezeichnet, die Geschwindigkeit in rot und die Beschleunigung in blau.

trapezoid Die Geschwindigkeit ist stückweise linear, jedoch stetig, wohingegen die stückweise konstante Beschleunigung Sprünge aufweist.	
sin² Die Knicke im Geschwindigkeitsprofil werden geglättet (durch Verwendung der Funktion sin² anstelle von Geraden), so dass die Sprünge in der Beschleunigung vermieden werden. Da der Bewegungsablauf klar definiert ist, kann der Ruck in der Regel nicht vom Anwender beschränkt werden. Seine Grenze für den Ruck hat nur dann Gültigkeit, wenn der Antrieb zu Anfang eine Beschleunigung ungleich 0 besitzt, Dann wird unter Berücksichtigung der Ruck-Begrenzung der Antrieb auf 0 heruntergefahren, bevor mit der aktuellen Bewegung begonnen wird. Verglichen mit dem trapezförmigen Geschwindigkeitsprofil wird das Abbremsen hier mehr Zeit einnehmen.	

SoftMotion Gerät: Skalieren/Mapping

Diese Seite erlaubt die Anpassung von physikalischem Gerät und seinem Abbild innerhalb der Applikation:

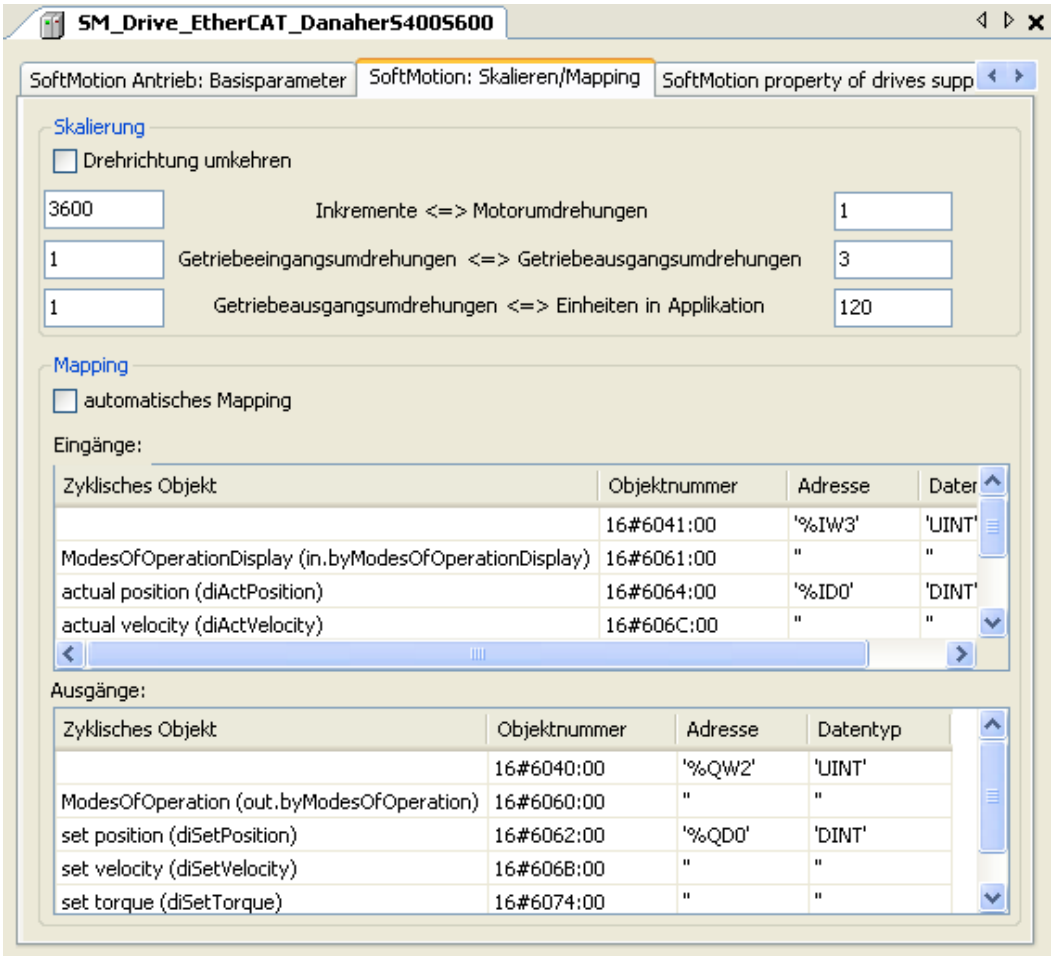


Abb. 505 Softmotion Geräte-Editor - Softmotion Gerät: Skalieren/Mapping

Skalierung

Hier muss der Faktor eingegeben werden, der die Umrechnung der ganzzahligen vom Gerät empfangenen oder zum Gerät gesandten Positionswerte (linker EIntrag in erster Zeile) in die vom IEC Programm verwendeten technischen Einheiten (rechter Eintrag in der dritten Zeile) bestimmt. Desweiteren kann durch Einstellung der weiteren Parameter ein Getriebe einbezogen werden.
In der Beispielskonfiguration wird ein Antrieb, der 3600 Inkremente für eine Rotation veranschlagt, so skaliert, dass die technischen Einheiten der Applikation gerade Winkelgrade sind.

Mapping

Bei Benutzung des **automatisches Mapping** werden die den Antrieb betreffenden IEC Parameter automatisch auf die entsprechenden Ein-/ und Ausgänge des Gerätes abgebildet. Nach Deaktivierung des Kontrollkästchens kann die Abbildung von Hand bearbeitet werden: ändern Sie hierzu Adresse oder Typ der Ein- und Ausgänge in der angezeigten Parameterliste, die gemäß der Gerätebeschreibungsdatei erstellt wurde.

Status

Hier wird der aktuelle Status des SoftMotion Geräts angezeigt.

Information

Diese Seite enthält aus der Gerätebeschreibungsdatei gelesene Information über das spezifische Gerät.

14.5.4 Konfiguration Drive Logical

Das SoftMotion Drive Logical ist ein virtueller Antrieb unter einem Master-Gerät oder einem Encoder.

Die Achse des logischen Antriebs spiegelt die Bewegung ihrer Master-Achse wieder, während sie eine eigenständige Instanz darstellt. Dies ermöglicht das Ausführen folgender Instruktionen auf ihr, ohne die Master-Achse zu beeinflussen: MC_SetPosition, MC_TouchProbe (darf auf logischen Achsen verwendet werden, wenn die Vorbedingung gilt, dass der Messtaster-Kanal nicht von der Master-Achse oder einer anderen logischen Achse verwendet wird,) und MC_AbortTrigger.

Der Achsentyp AXIS_REF_LOGICAL_SM3 ist Teil der Bibliothek SM3_Basic. Die Instanz dieses Typs wird mit Hilfe des SoftMotion-Drive-Logical-Editors konfiguriert:

- SoftMotion Drive: Logical
- SM_Drive_Logical: Konfiguration
- SM_Drive_Logical: I/O Abbild
- Status
- Information

SoftMotion Drive: Logical

Der erste Reiter des SoftMotion-Drive-Editors stellt Eingabefelder für die Konfiguration zur Verfügung.

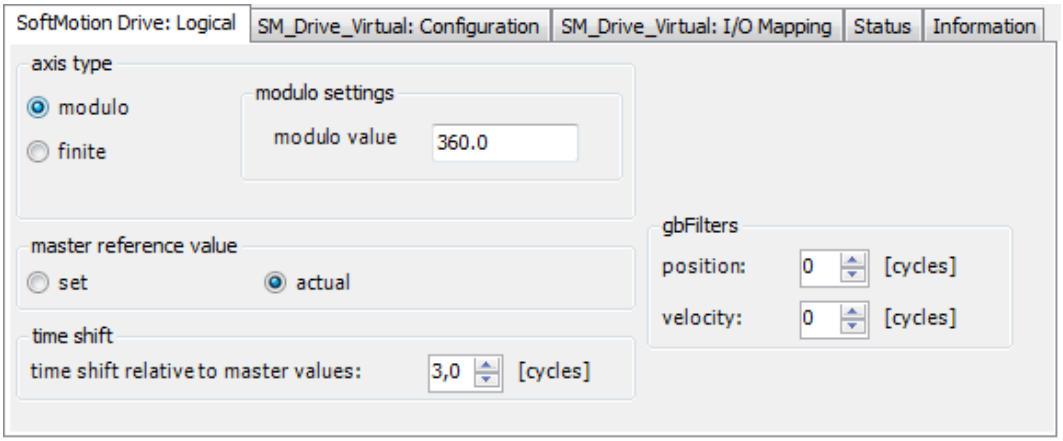


Abb. 506 SoftMotion Drive: Logical, Modulo-Achse

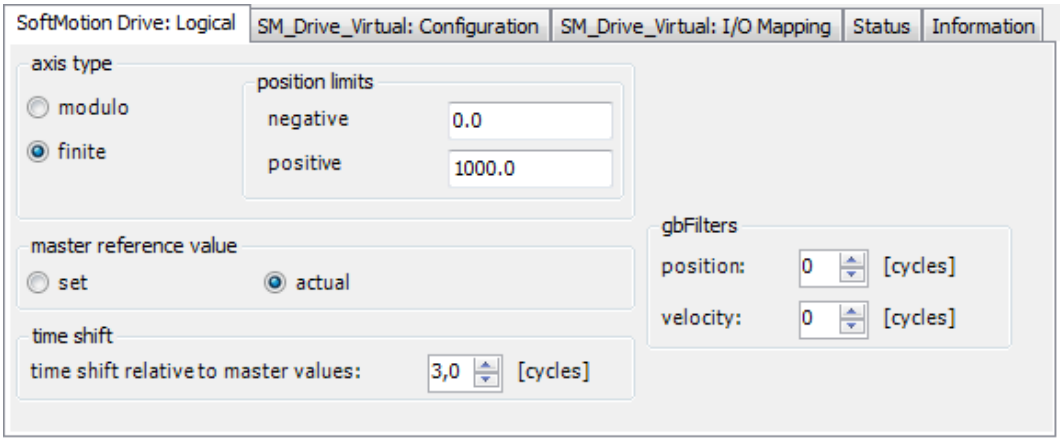


Abb. 507 SoftMotion Drive: Logical, Finite-Achse

Achsentyp:

Ist der Achsentyp **modulo**, dann ist das Eingabefeld **modulo settings** sichtbar, mit der Möglichkeit den Modulo-Wert zu setzen.

Ist der Achsentyp **finit**, dann kann der negative und positive Positionsgrenzwert gesetzt werden.

master reference value:

Hier kann im Optionsfeld zwischen Soll- und Istwert des Master-Referenz-Werts gewählt werden. Abhängig von dieser Eingabe wird entweder der Sollwert oder der Istwert des Masters als Eingang für die logische Achse und der Totzeit-Kompensation verwendet.

time shift:

Hier kann die Anzahl der Zyklen gesetzt werden, die extrapoliert werden soll. Die Anzahl der Zyklen legt die Zeitspanne fest, um die die Sollwerte extrapoliert werden müssen.

input data filter:

Der Filter berechnet den gleitenden Durchschnitt der Werte. Abhängig von der gewählten Anzahl an Zyklen werden die zugehörigen Werte für die Berechnung des Durchschnitts der Position bzw. der Geschwindigkeit verwendet. Das Ergebnis sind geglättete Werte. Der Bereich geht von 0 bis 9. Ist die Anzahl der Zyklen auf 0 gesetzt, wird der letzte Wert als gefilterte Position oder gefilterte Geschwindigkeit verwendet. Ansonsten wird der berechnete Wert verwendet.

SM_Drive_Logical: Konfiguration

Hier ist die Instanz der Datenstruktur des logischen Antriebs mit den konfigurierten Einstellungen in einer Tabelle aufgelistet.

SM_Drive_Logical: I/O Abbild

Mapping ist nicht möglich.

Status

Hier ist der Status des logischen Antriebs sichtbar.

Information

Diese Seite enthält Informationen aus der Gerätebeschreibungsdatei über das spezifische Gerät.

14.5.5 Übersicht über spezielle Anwendungsfälle

Die Bibliothek SM3_Basic.library enthält einige übergeordnete Funktionsblöcke, die mit dem Antrieb nicht über die standardmäßige Soll-/Ist-Wert-Schnittstelle interagieren, sondern durch Lesen/Schreiben der Parameter von AXIS_REF Kommandos geben oder Einstellungen des Antriebs verändern. Durch Anwenden dieser Funktionsblöcke können folgende Anwendungsfälle geeignet bearbeitet werden:

Reinitialisierung eines Antriebs

Eine Änderung der Antriebsparameter (wie Skalierungsfaktoren, Positionszyklen von rotatorischen Antrieben etc.) während der Laufzeit erfordert eine Reinitialisierung des Antriebs. Durch Aufruf des Funktionsblocks SMC3_ReinitDrive mit einer steigenden Flanke am Eingang Execute wird der Parameter wCommuncationState des Antriebs zurückgesetzt und der komplette Start-up Prozess des Antriebs wiederholt.

Ein-/Abschalten eines Antriebs

Die AXIS_REF Parameter bRegulatorOn und bDriveStart (Zugriff über Funktionsblock block MC_Power) bestimmen Leistungsstufe und Abbremsen eines Antriebs.

Zur Endstufenfreigabe muss der Parameter bRegulatorOn gesetzt werden; in bRegulatorRealState meldet der Antrieb den momentanen Status dieses Prozesses. Sofern vom Antrieb unterstützt, setzt ein Reset von bRegulatorOn während einer Bewegung das Drehmoment mit sofortiger Wirkung außer Kraft. Der Funktionsblock meldet in diesem Fall einen Fehler und setzt die Achsen in den Zustand "errorstop".

Die Variable bDriveStart bestimmt, ob sich der Antrieb bewegen darf oder nicht; in bDriveStartRealState steht der aktuelle Zustand dieses Parameters.

Sofern vom Antrieb unterstützt, führt ein Rest der Parameters bDriveStart während der Bewegung zum Abbremsen des Antriebs (QuickStop) gemäß der konfigurierten Rampe; der Antrieb wird danach die dabei erreichte Position beibehalten. Solange wie der QuickStop Mechanismus aktiv ist, bleibt die Achse im Zustand “stopping”. Falls der Antrieb zuvor von einem Funktionsblock bewegt wurde, zeigt dieser FB nun "CommandAborted". Das gilt mit Ausnahme des FBs MC_Stop, welcher einen Fehler melden und die Achsen in den Zustand "errorstop" setzen wird.

Handhabung mechanischer Bremsen

Sofern vom Antrieb unterstützt, kann die Variable eBrakeControl mit Hilfe des Funktionsblocks SMC3_BrakeControl (zurück)gesetzt werden. Drei unterschiedliche Modi sind möglich (SMC3_BrakeSetState: SMC_BRAKE_AUTO, SMC_BRAKE_OPEN, SMC_BRAKE_CLOSE).

Die Variable bBrakeClosedRealState meldet den aktuellen Status der Bremse.

Vom SoftMotion Standpunkt aus hängen Kontrolle wie auch Zustand der mechanischen Bremse weder vom Zustand des Antriebs noch von der laufenden Bewegung ab.

Wechsel zwischen den Kontrollarten

Der Funktionsblock SMC_SetControllerMode erlaubt dem Anwender ein Wechsel zwischen den verschiedenen Kontrollarten. Die Parameter byControllerMode bzw. byRealControllerMode von AXIS_REF repräsentieren den gewünschten bzw. den aktuellen Zustand des Reglers. Mögliche Wert sind:

1	Moment/-Kraft-Regelung
2	Geschwindigkeitsregelung
3,0	Lageregelung
4	Stromregelung

Nach Schreiben der Variablen byControllerMode wird der aktuelle Zustand solange geprüft bis er mit dem gewünschten Kontrollmodus übereinstimmt.

Referenzpfad (antriebsgesteuert)

SoftMotion bietet zwei Möglichkeiten eine Antrieb zu referenzieren: eine vom Antrieb, eine vom Regler gesteuerte.

Letztere wird durch Aufruf des Funktionblocks SMC_Homing ausgeführt. Dabei muss das Referenzsignal Teil des I/O Abbilds des Reglers sein. Während des Referenzierens bleibt der Antrieb im Kontrollmodus "Lageregelung".
Im allgemeinen ist diese Methode nicht so genau, da die Erfassung der Position (beim Berühren des Referenz-Schalter) auf den vom Antrieb erhaltenen aktuellen Positionen basiert; folglich ist die Genauigkeit durch den Kommunikationszyklus beschränkt.

Im Gegensatz dazu ist beim antriebsgesteuerten Referenzpfad der Referenz-Schalter zum Antrieb selbst verdrahtet; somit vermag er die Position schneller und mit höherer Genauigkeit auszuwerten. Während des antriebsgesteuerten Referenzpfads ist der Antrieb in einem anderen Kontrollmodus, in welchem er nicht die Sollwerte des Reglers befolgt, sondern durch interne Prozesse gesteuert wird. Sobald der Referenzpfad beendet wird, wird der Antrieb in seinen ursprünglichen Kontrollmodus zurückversetzt.

Der Funktionsblock MC_Home startet den antriebsgesteuerten Referenzpfad durch Setzen von fReference and bStartReference. Zusätzlich wird der Zustand von AXIS_REF (wState) auf “homing” gesetzt. Sobald der Antrieb in den Referenzpfad übergeht (d.h. dass er die vom Regler vorgegebenen Werte nicht mehr beachtet), setzt der Antrieb den Parameter auf TRUE. Um das Ende des Referenzpfads anzuzeigen setzt der Treiber die Werte von bStartReference und bStartReferenceRealState wiederum auf FALSE .

Positionserfassung (Latching; antriebsgesteuert)

SoftMotion bietet zwei Möglichkeiten die aktuelle Position eines Antriebs zu erfassen: bei der genaueren wird das Signal, welches den Antrieb veranlasst seine Position zu erfassen, direkt mit dem Antrieb verdrahtet. Die zweite Möglichkeit benutzt einen Reglereingang zum Messen der aktuelle Position. Die erste Methode ist zwar präziser (da es weniger Verzögerung bei der Übertragung des Signals gibt), benötigt aber einen Antrieb der diese Möglichkeit vorsieht. Außerdem wird die Positionserfassung innerhalb eines Fensters i.a. nur von der zweiten Methode unterstützt. Gleichwohl ist die Erfassung im Antrieb die Standardmethode.

Der zur Positionserfassung bereitgestellte Funktionsblock MC_TouchProbe kommuniziert mit dem Antrieb über die Datenstruktur aCaptDesc, welche Teil der AXIS_REF Parameter ist); diese Datenstruktur beinhaltet alle Informationen über erfasste Positionen, Start und Ende der Erfassung und über gültige Positionen des Erfassungsfensters.

14.5.6 Beispielapplikationen

Übersicht

Bitte beachten Sie, dass »PLC Designer« SoftMotion ein kosten- und lizenzpflichtiges Produkt ist, das nicht auf allen »PLC Designer«-kompatiblen Steuerungen freigeschaltet ist. Fragen Sie bitte den Hersteller Ihrer Steuerung, ob das für Ihr Gerät der Fall ist.

Verwenden Sie »PLC Designer« SP WinV3 oder »PLC Designer« SP RTE, dann achten Sie bitte darauf, dass Sie die SoftMotion-Variante des Laufzeitsystems gestartet haben (im Fall von »PLC Designer« SP WinV3 finden Sie diese im Startmenü->Programme->PLC Designer->PLC Designer SP Win V3->PLC Designer SP SoftMotion Win V3 als Test-Version) bzw. benötigen ein spezielles Setup im Fall der »PLC Designer« SP SoftMotion RTE.

Die in diesem Kapitel beschriebenen Beispiele verwenden alle das Laufzeitsystem »PLC Designer« SP SoftMotion Win V3. Sie können aber bei Bedarf mittels „Gerät aktualisieren“ auf einen SoftMotion-Controller anderen Typs umgestellt werden. Außerdem kommen in den Beispiel virtuelle Antriebe zum Einsatz. Auch diese können durch reale gleichen Namens ersetzt werden.

Sehen Sie folgende Programmierbeispiele:

Bewegungssteuerung einzelner Achsen

Bewegungssteuerung einzelner Achsen in CFC mit Visualisierungs-Template

Kurvenscheiben-Antriebssteuerung mit Hilfe einer virtuellen Zeitachse

Wechselnde Kurvenscheiben

Lageregelung im Controller (SM_Drive_PosControl)

CNC-Beispiel 1: Direktes Erzeugen der OutQueue

CNC-Beispiel 2: Dekodierung online mit Verwendung von Variablen

CNC-Beispiel 3: Bahnvorverarbeitung online

CNC-Beispiel 4: CNC mittels Tabelleneditor

CNC-Beispiel 5: CNC aus Datei

Bewegungssteuerung einzelner Achsen

(Siehe hierzu das Beispielprojekt PLCopenSingle.project)

Dieses Beispiel demonstriert, wie ein Antrieb mittels der nach PLCopen standardisierten Funktionsblöcke gesteuert werden kann.

Dazu wurde im Gerätebaum unter **SoftMotion General Drive Pool** ein virtueller Antrieb eingefügt

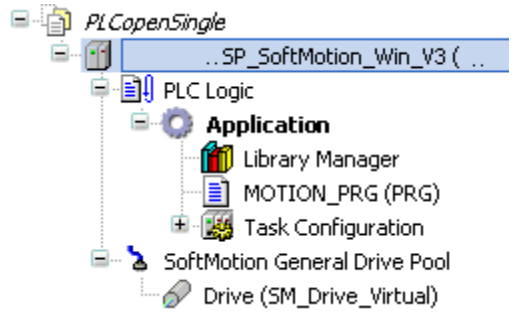


Abb. 508 Gerätebaum, Antrieb eingefügt

Durch Doppelklick auf das neue Gerät wird der zugehörige Konfigurationsdialog geöffnet:

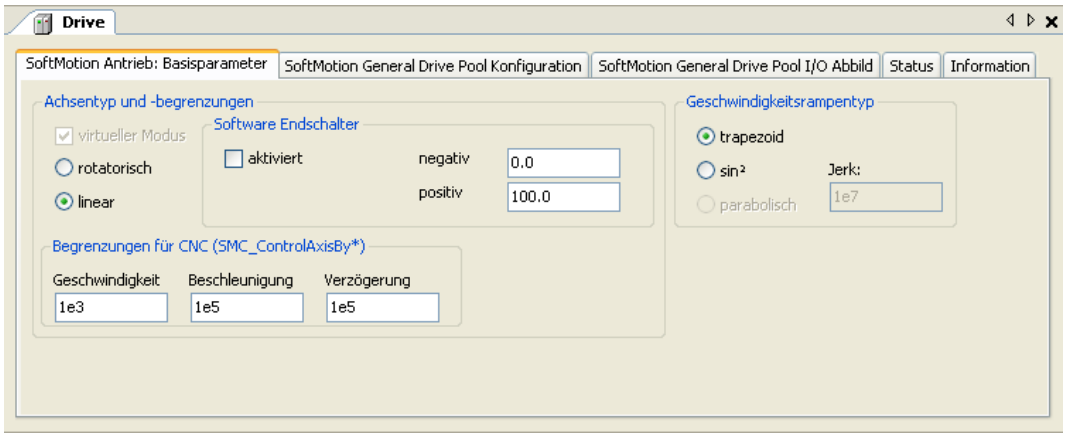


Abb. 509 SoftMotion Antrieb: Basisparameter, Dialog im Offline-Modus

In der **Taskkonfiguration** rufe man das Programm **MOTION_PRG** auf, das eine Bewegung dieser Achse erzeugen wird.

Dieses Programm soll im folgenden erstellt werden:

Wir legen ein Programm in der Sprache **ST** mit dem Namen MOTION_PRG an und deklarieren darin folgende Variablen:

```
PROGRAM MOTION_PRG
VAR
iStatus: INT;
Power: MC_Power;
MoveAbsolute: MC_MoveAbsolute;
p:REAL:=100;
END_VAR
```

Wir programmieren eine Zustandsmaschine, die im ersten Zustand die Achse in Regelung schaltet:

```
CASE iStatus OF
0:
Power(Enable:=TRUE, bRegulatorOn:=TRUE, bDriveStart:=TRUE, Axis:=Drive);
IF Power.Status THEN
iStatus := iStatus + 1;
END_IF
```

Im nächsten Zustand wird die Achse durch den Baustein MC_MoveAbsolute auf eine Position p gefahren:

```
... (* continuation of the program shown above *)
1:
MoveAbsolute(Execute:=TRUE, Position:= p, Velocity:=100, Acceleration:=100,
Deceleration:=100, Axis:=Drive);
IF MoveAbsolute.Done THEN
MoveAbsolute(Execute:=FALSE, Axis:=Drive);
iStatus := iStatus + 1;
END_IF
```

Im letzten Schritt wird sie wieder auf Position 0 zurückbewegt:

```
... (* continuation of the program shown above *)
2:
MoveAbsolute(Execute:=TRUE, Position:= 0, Velocity:=100, Acceleration:=100,
Deceleration:=100, Axis:=Drive);
IF MoveAbsolute.Done THEN
MoveAbsolute(Execute:=FALSE, Axis:=Drive);
iStatus := 1;
END_IF
```

Anschließend wird in den Zustand 1 zurückgesprungen. Dadurch wird man also eine Pendelbewegung zwischen Position p und 0 erhalten.

Wir können das Projekt übersetzen auf die Steuerung laden und starten, sofern wir diese gestartet und im Kommunikationsdialog die korrekte Adresse eingestellt haben.

Öffnen wir das Gerät **Drive**, während wir auf der Steuerung **eingeloggt** sind, so sehen wir jetzt die Bewegung der Achse:

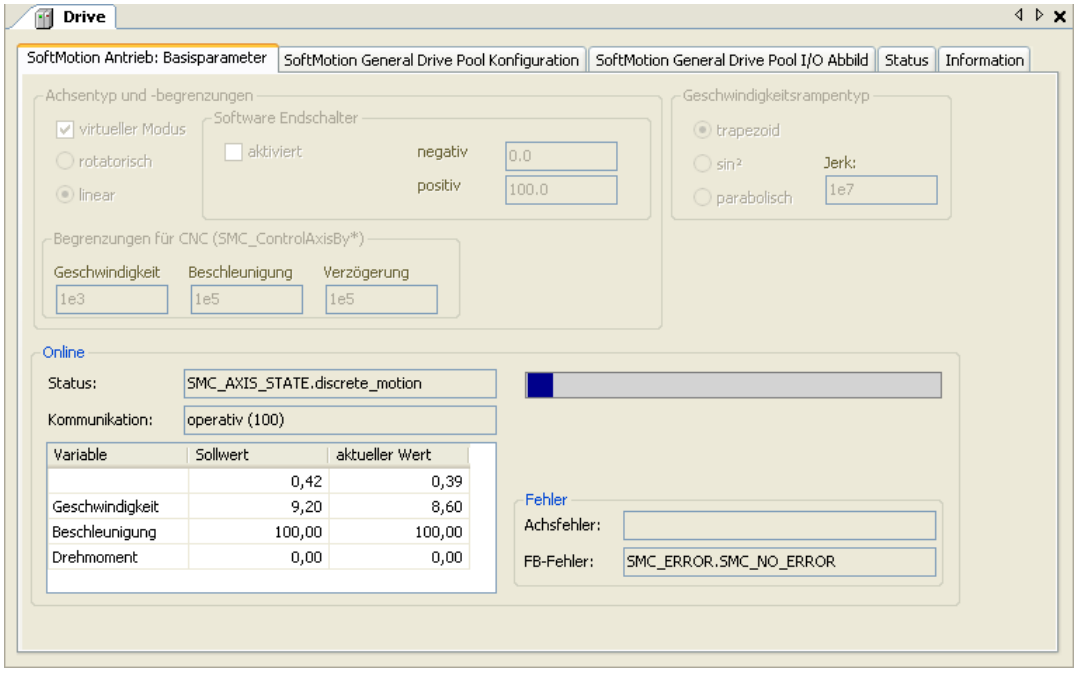


Abb. 510 SoftMotion Antrieb: Basisparameter, Dialog im Online-Modus

Bewegungssteuerung einzelner Achsen in CFC mit Visualisierungs-Template

(Siehe hierzu das Beispielprojekt PLCopenSingle2.project)

Anstelle von ST kann auch jede andere IEC-Sprache als Programmiersprache dienen, wie CFC im vorliegenden Beispiel. Dieses dient dazu, den Start- und Unterbrechungsmechanismus der Funktionsbausteine nachzuvollziehen. Außerdem können die verschiedenen Startmodi für den Baustein MC_MoveAbsolute für rotatorische Achsen ausprobiert werden.

Konfigurieren Sie die **Taskkonfiguration** und fügen im Gerätebaum einen **Antrieb** ein, wie im Beispiel Bewegungssteuerung einzelner Achsen beschrieben, nur verwenden Sie dieses Mal einen rotatorischen Antrieb mit der Periode 360. Das Programm MOTION_PRG wird in **CFC** geschrieben und beinhaltet lediglich drei Aufrufe von Instanzen der Bausteine **MC_Power** (notwendig, um die Achse zu aktivieren),

MC_MoveAbsolute und **MC_MoveVelocity**:

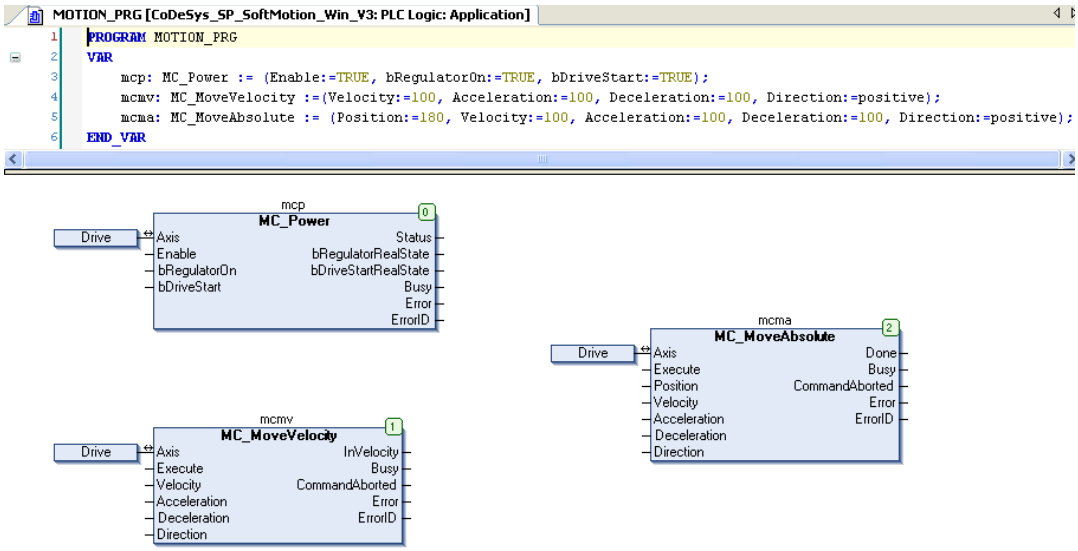


Abb. 511 Programm MOTION_PRG

Die Initialisierung der Bausteineingänge empfiehlt sich, damit wir später beim Starten dieser Testapplikation die Werte nicht stets von Neuem eingeben müssen.

Außerdem erstellen wir eine **Bedien-Visualisierung**, wozu wir die in den Bibliotheken vorhandenen **Baustein-Templates** verwenden und über das Frame-Konzept mit den Bausteininstanzen verknüpfen:

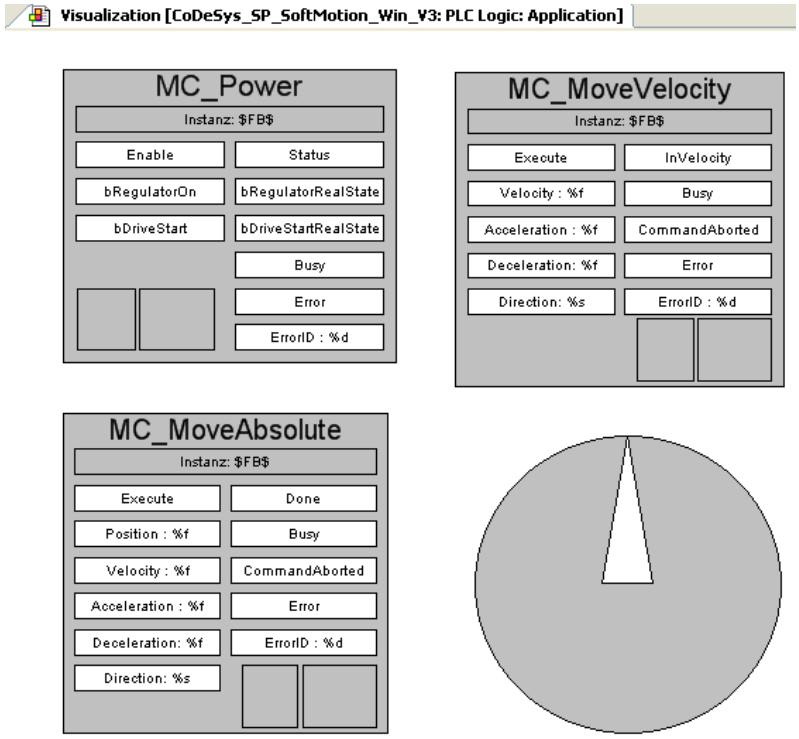


Abb. 512 Visualisierung, Frames mit Baustein-Templates

Nun können wir das Projekt fehlerfrei **übersetzen**, uns auf die Steuerung **einloggen** und **starten**. Durch Drücken des Execute-Eingangs von **MoveVelocity** beginnt sich der Antrieb zu drehen. Drücken vom Execute des **MoveAbsolute** positioniert den Antrieb auf die eingestellte Position, wobei entsprechend der Angabe **Direction: positive** nur in positive Richtung gedreht wird. Der Baustein MoveVelocity wird dadurch unterbrochen.

Es empfiehlt sich, mit diesen Bausteinen zu spielen, verschiedene Geschwindigkeiten und Beschleunigungen auszuprobieren (durch direktes Editieren innerhalb der Module der Visualisierung) und die Richtungsmodi (positive/negative/current/shortest/fastest) von MoveAbsolute zu testen.

Kurvenscheiben-Antriebssteuerung mit Hilfe einer virtuellen Zeitachse

(Siehe hierzu das Beispielprojekt PLCopenMulti.project)

Das folgende Beispiel demonstriert, wie eine **periodische Kurvenscheibe** auf einem **linearen Antrieb** umgesetzt werden kann. Zusätzlich wird die **Nockenfunktion** demonstriert.

1. Dazu legt man im **Gerätebaum** eine beliebige periodische **Kurvenscheibe** an, die sich auf eine Masterachse zwischen 0 und 10 sec bezieht, und mindestens eine invertierende **Nocke** mit ID1 enthält.

Beispiel:

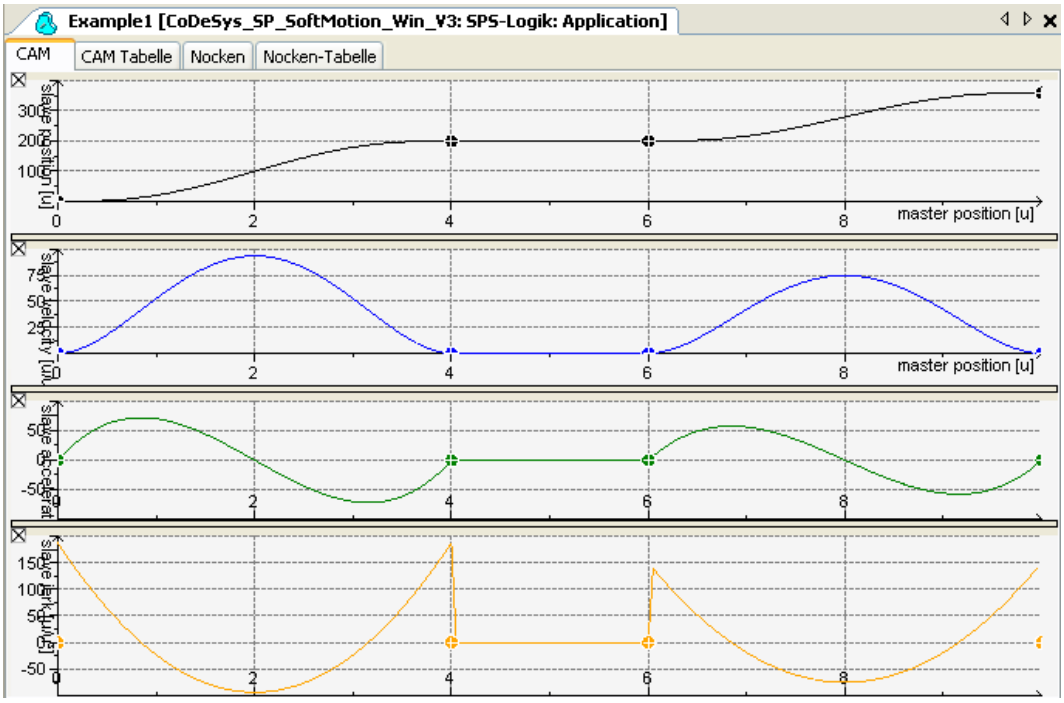


Abb. 513 Kurvenscheibe "Example", CAM Konfigurationsdialog

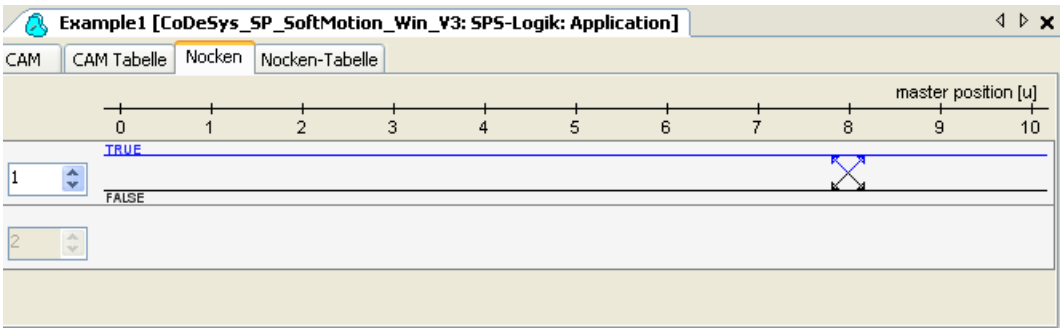


Abb. 514 CAM "Example", Tappets configuration

2. Im Gerätebaum werden zwei **modulare Antriebe** mit den Namen "Drive" (Periode 360) und "Virtual“ (Periode 10) definiert:

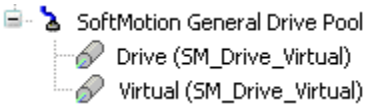


Abb. 515 Gerätebaum mit zwei modularen Antrieben

3. Das Programm **MOTION_PRG** ist in **FUP** erstellt und beinhaltet die Aufrufe folgender Bausteine:

```
PROGRAM MOTION_PRG
VAR
power1, power2: MC_Power;
TableSelect: MC_CamTableSelect;
CamIn: MC_CamIn;
Tappet: SMC_GetTappetValue;
MoveVirtual: MC_MoveVelocity;
END_VAR
```

Nach den **Power**-Bausteinen (Power1, Power2) für die Achsen, wird zunächst der **MoveVelocity**-Baustein aufgerufen, der den virtuellen Master bewegt. **TableSelect** wählt die gewünschte Kurvenscheibe aus, und **CamIn** setzt diese um. Der Baustein **Tappet** überprüft die Stellung des Nockenschalters. Da dieser invertiert programmiert wurde, wird er alle 10 Sekunden umschalten.

Nun kann das Programm übersetzt und auf der Steuerung gestartet werden.

Um die Soll- bzw. Istposition zu kontrollieren, erstelle man wieder eine **Visualisierung**, mit deren Hilfe man die einzelnen Bausteine und die Position der Achsen überprüfen kann.

Wechselnde Kurvenscheiben

(Siehe hierzu das Beispielprojekt PLCopenMultiCAM.project)

Dieses Beispiel zeigt, wie eine Kurvenscheiben-Bewegung mit **zwei sich abwechselnden Kurvenscheiben** realisiert werden kann. Es wurde in ST programmiert und führt die selben Aktionen wie das Beispiel Kurvenscheiben-Antriebssteuerung mit Hilfe einer virtuellen Zeitachse aus. Am Ende der ersten Kurvenscheibe setz der MC_CamIn-Baustein den Ausgang bEndOfProfile, aufgrund dessen der jeweils andere **MC_CamTableSelect** zur Anwendung kommt und **MC_CamIn** neu gestartet wird.

Lageregelung auf der Steuerung (SM_Drive_PosControl)

Sehen Sie hierzu das Beispielprojekt **PosControl.project**, das im Lieferumfang enthalten ist.

In diesem Beispiel wird nicht, wie sonst üblich, ein Servo-Regler verwendet, der neben Strom- und Drehzahlregelung auch die Lageregelung übernimmt, sondern stattdessen wird gezeigt, wie ein drehzahlgesteuertes Gerät (z.B. Frequenzumrichter) mit Positionsrückführung von »PLC Designer« lagegeregelt wird.

Voraussetzung ist, dass ein Gerät zur Verfügung steht, das über Sollgeschwindigkeit angesteuert wird und das seine aktuelle Position zurück liefert. Als exemplarisches Beispiel wird in diesem Beispiel eine 10V-Analog-Ausgangsklemme (Beckhoff EL4031) verwendet (deren Signal als Drehzahlsollwert für einen Frequenzumrichter verwendet wird) und für die Lagerückführung eine Encoder-Klemme (Beckhoff EL5101).

Anleitung

Die Klemmen sind im Gerätebaum anzulegen und unter SoftMotion General Drive Pool ist eine lagegeregelte Achse (SM_Drive_PosControl) anzufügen.

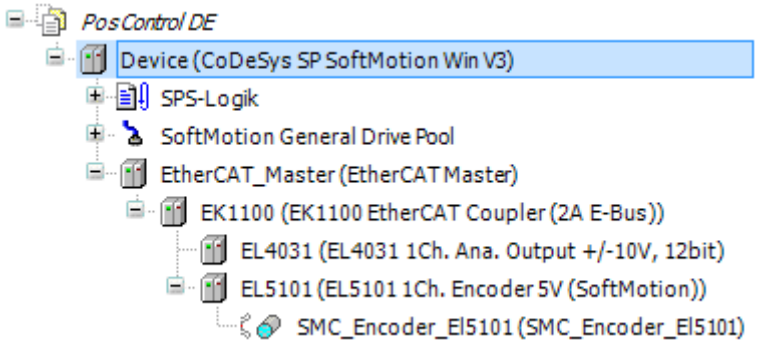


Abb. 516 Gerätebaum

Mit Öffnen des Geräts **SM_Drive_PosControl** sind die gewöhnlichen Einstellungen in den Registerkarten **Basisparameter** und **Skalieren/Mapping** einzutragen. Dabei muss für die richtige Konfiguration der Skalierung bekannt sein, wie viele Inkremente der Encoder-Klemme einer Motor-Umdrehung entsprechen. In diesem Beispiel sind 4096 Inkremente eine Umdrehung des Motors und – es wird in Winkelgrad gearbeitet - damit 360 Applikationseinheiten.

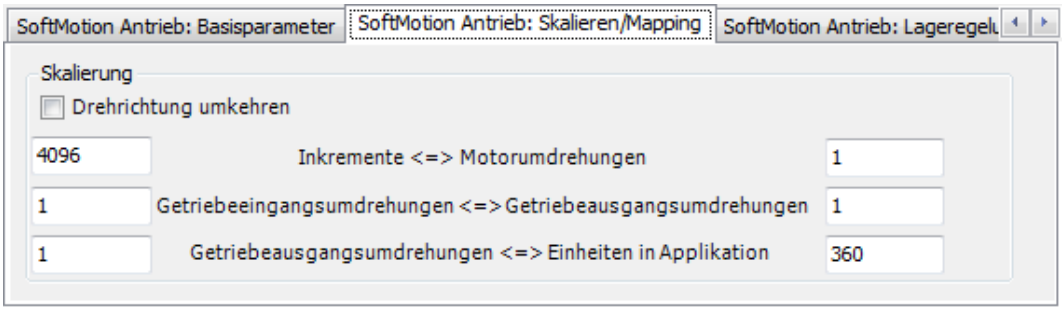


Abb. 517 SoftMotion Antrieb: Skalieren/Mappping

Anschließend werden in der Registerkarte **Position Control Loop** folgende Einstellungen gesetzt.

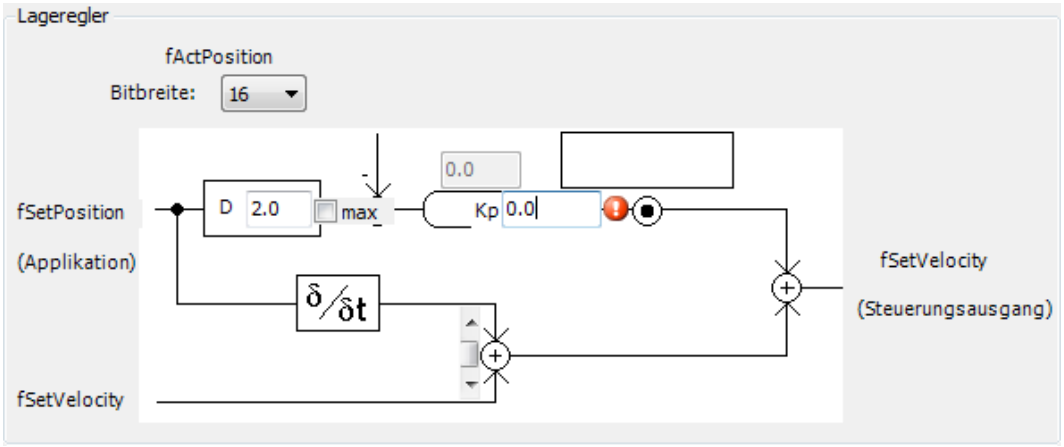


Abb. 518 SoftMotion Antrieb: Lageregler

D

Abb. 519

Totzeit legt die Anzahl an Zyklen fest, um welche die empfangene Istposition (Encoder) phasenverschoben zur Sollposition unserer Achse ist. Diese ist abhängig von den verwendeten Komponenten und muss experimentell ermittelt werden (s.u.).

Kp

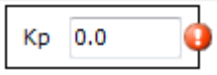
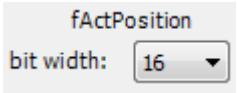


Abb. 520

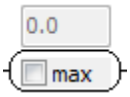
Die Proportianalitätskonstante ist der Faktor, mit welchem der Lagefehler (Abweichung zwischen Soll- und Istposition) multipliziert wird, um später auf die Sollgeschwindigkeit addiert zu werden. Diesen setzen wir zunächst auf 0 und werden ihn experimentell ermitteln (s.u.)

Bitbreite des Istwerts



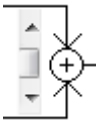
Die Bitbreite des Istwerts wird in Abhängigkeit von den verwendeten Komponenten empfangen und kann als 16-, 24- oder 32-Bit Wert gesetzt sein. Dies wird dem Regelbaustein hier mitgeteilt. Hier ist der Wert 16 zu setzen, da die verwendete Komponente die Position als UINT übergibt.

Lagefehler-Überwachung



Die Lagefehler-Überwachung bleibt zunächst ausgeschaltet; kann aber anschließend bei Bedarf wieder einschaltet werden. Dazu wird die Checkbox aktiviert und ein maximal zulässiger Schleppfehler eingegeben. Wird dieser während des Betriebs überschritten, geht die Achse in einen Fehlerzustand.

fSetVelocity



Da in der Regel Sollposition und Sollgeschwindigkeit zueinander stimmig berechnet werden, ist dieses Feld von geringerer Bedeutung und muss nur im speziellen Fall verwendet werden. Deshalb wird der Regler komplett nach unten geschoben, dann wird die Geschwindigkeit als Quelle verwendet.

Nun muss noch die Wichtung der Geschwindigkeitswerte eingestellt werden, die an den Aktuator gesendet wird. Dazu muss die Maximal-Geschwindigkeit in Applikationseinheiten und der dazugehörende Rohwert der übertragenen Daten bekannt sein. In diesem Beispiel wird die Maximalgeschwindigkeit durch Ausgabe des Wertes 16#7FFF erreicht, welches einer Geschwindigkeit von 10 Umdrehungen / s entspricht, d.h. gemäß den gemachten Einstellungen entspricht es 3600 Grad / s.

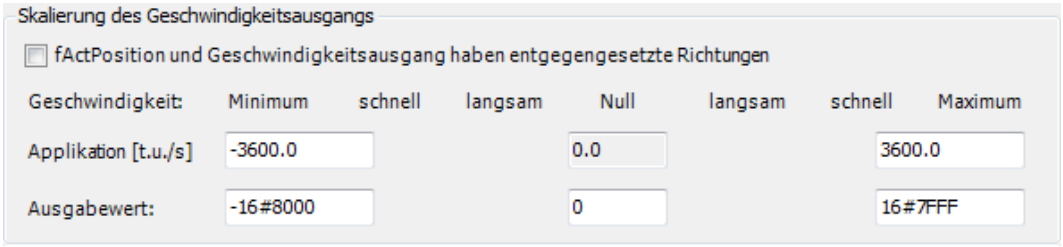


Abb. 521 SoftMotion Antrieb: Lageregler, Skalierung des Geschwindigkeitsausgangs

Im nächsten Schritt wird die Verbindung zwischen Daten der Achse und der Geräte am Bus geschaffen. Die verfügbaren zyklischen Daten der Achse liegen in den Datenstrukturen **in** und **out**. Diese können entweder programmatisch verbunden werden, oder gleich direkt im Geräteeditor des Ein- bzw. Ausgabegeräts. Um also den Ausgang, die Solldrehzahl, zur Klemme weiterzuleiten, öffnen wir das Gerät EL4031 und ordnen unter **EtherCAT I/O Abbild** dem Ausgang die Variable **out.iSetVelocity** (im Falle eines 32-Bit Ausgangs wird **out.diSetVelocity** verwendet) der Achse zu.

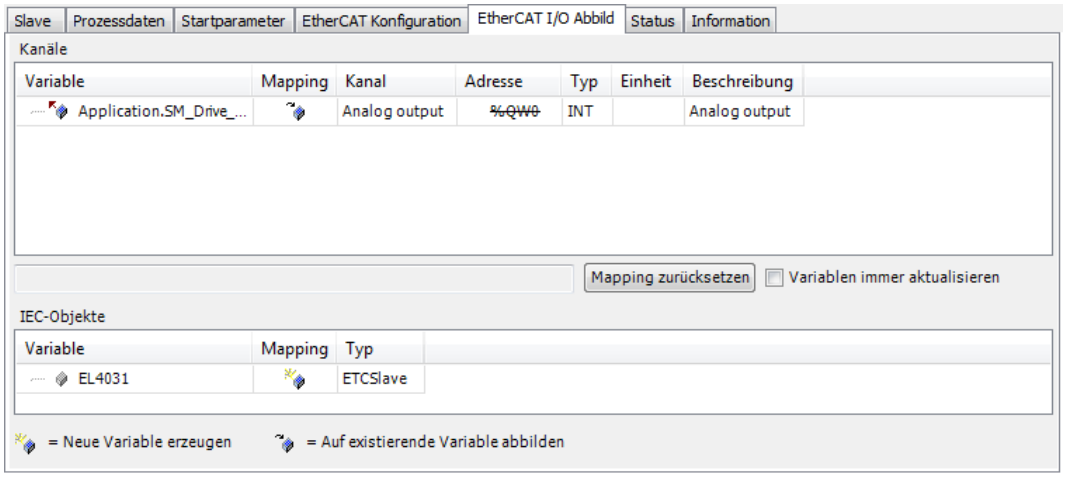


Abb. 522 Gerät EL4031: EtherCAT I/O Abbild

Ebenso wird beim Lageeingang verfahren. Mit Öffnen von EL5101 wird der Positionseingangswert auf **in.wActPosition** festgelegt,im Falle eines 32-Bit Eingangs wird auf **in.dwActPosition** gesetzt.

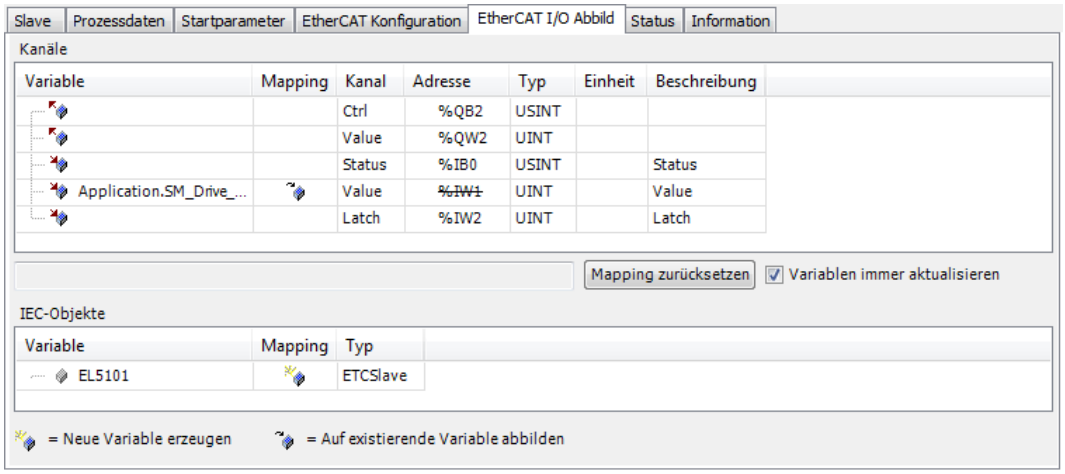


Abb. 523 Gerät EL5101: EtherCAT I/O Abbild

Damit die Achse funktioniert, müssen mindestens die in der Tabelle angegebenen Werte der Eingangsdaten gesetzt werden. Falls die verwendeten Komponenten diese Daten anbieten (bei z.B. einem Frequenzumrichter mit Feldbusanschaltung ist dies häufig der Fall), werden diese eben aus den Eingangsdaten bezogen. Hier, da dieses einfache Interface der Daten nicht zur Verfügung gestellt wird, werden sie manuell in unserer Applikation gesetzt.

SM_Drive_PosControl.in.bDriveStartRealState := TRUE;

SM_Drive_PosControl.in.bRegulatorRealState := TRUE;

Folgende Eingänge...

bLimitPos	BOOL	Endschalter in positive Richtung (nur bei finite Achsen; zur Aktivierung bHW
bLimitNeg	BOOL	Endschalter in negativer Richtung
wActPosition/ dwActPosition	WORD/ DWORD	Istposition
bExternalError	BOOL	Externer Fehler
bRegulatorRealState	BOOL	TRUE: Achse in Regelung
bDriveStartRealState	BOOL	FALSE: Achse in Schnellhalt (Quick Stop)

... und Ausgänge stehen zur Verfügung:

bRegulatorOn	BOOL	TRUE: Achse soll in Regelung schalten
bDriveStart	BOOL	FALSE: Achse soll Schnellhalt ausführen
diSetVelocity / iSetVelocity	DINT/INT	Sollgeschwindigkeit

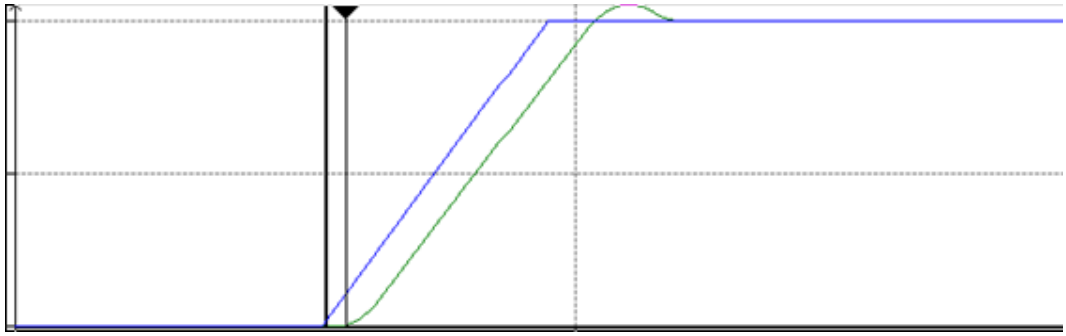
Jetzt kann die Applikation online gehen und die Einstellung der Reglerparameter vorgenommen werden.

Hinweis: Bitte beachten Sie, dass die Achse sich dabei evtl. unkontrolliert bewegen kann und treffen Sie entsprechende Sicherheitsvorkehrungen.

Zunächst wird versucht, die Achse ohne Lageregler zu betreiben (fKp ist bereits auf 0.0 gestellt) und die Skalierungseinstellungen werden verifiziert. Dazu schalten wir die Achse mit **MC_Power** ein und starten **MC_MoveVelocity**. Es kann nun beobachtet werden, dass sich die Achse mit der programmierten Geschwindigkeit (1U/s) bewegt. Bei Abweichungen muss die Skalierung entsprechend korrigiert werden.

Die Bewegung (z.B. mit MC_MoveRealtive) wird nun beendet und die Trace-Funktion gestartet.

Um die Totzeit des Systems zu ermitteln wird der zeitlichen Versatz zwischen Sollposition und Istposition gemessen. Dazu wird in **MC_MoveRelative** die Maximalgeschwindigkeit und eine enorm große Beschleunigung (z.B. 108) eingestellt. Dann wird die Traceaufzeichnung gestartet und **MC_MoveRealtive**. Die Totzeit wird erhalten, indem die Zeitdifferenz zwischen dem Bewegungsstart der Sollposition und der ersten Reaktion der Istposition gemessen wird.



Diese absolute Zeit wird durch die Zykluszeit geteilt und bestimmt so die Anzahl der Versatz-Zyklen.

Jetzt wird versucht, die korrekte Einstellung für fKp zu finden. Dazu wird fKp auf eine kleine Zahl (z.B. 0.0001) gesetzt und dann schrittweise erhöht. Bei jeder Änderung muss mit der Traceaufzeichnung das Verhalten überprüft werden. Sobald Schwingungen feststellt werden, ist die Obergrenze erreicht und dann wird die endgültige Einstellung von fKp um etwa 10% reduziert. Diesen Wert wird in der Gerätekonfiguration eingetragen und nun kann die Achse verwendet werden.

CNC-Beispiel 1: Direktes Erzeugen der OutQueue

Beachten Sie das dem Beispiel entsprechende Projekt "CNCdirect.project", das im Lieferumfang enthalten ist.

1. Erstellen des NC-Programms im CNC-Editor:

Wir erstellen ein Beispielprogramm, welches sich zwischen x aus [0,100] und y aus [0,100] abspielt. Außerdem definieren wir Geschwindigkeiten und Beschleunigungen für die Bahn und setzen zwei Wegschaltpunkte auf der Bahn. Z.B.:

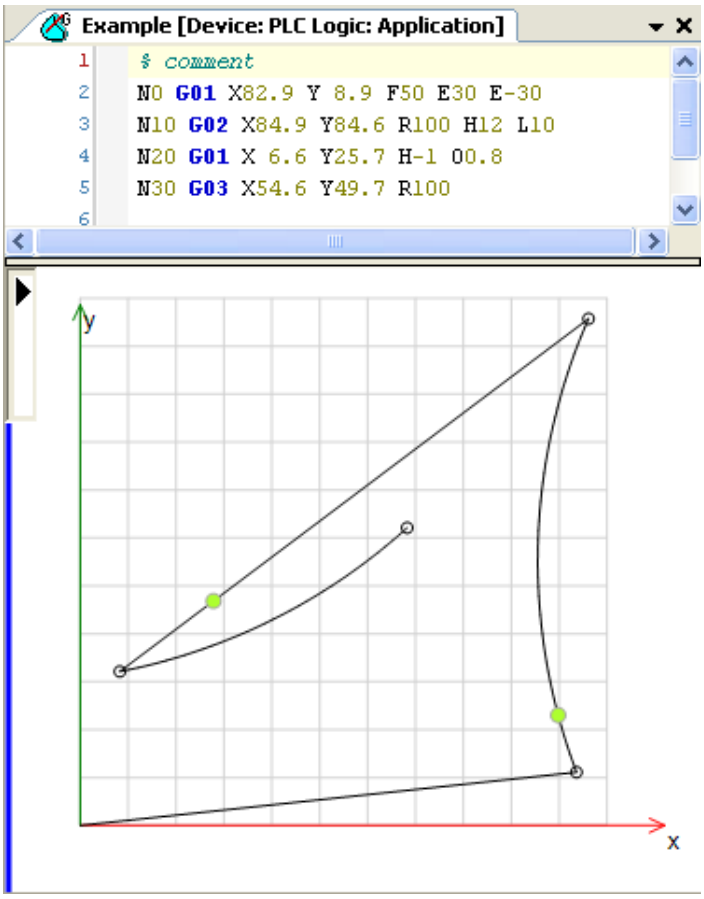


Abb. 524 CNC-Programm der Application CNCdirect

Als Übersetzungsmodus wählen wir SMC_OutQueue.

2. Drive Interface, Steuerungskonfiguration:

Eine Antriebsstruktur mit 2 linearen Antrieben wird definiert; dabei muss auch die maximale Geschwindigkeit etc. gesetzt werden.

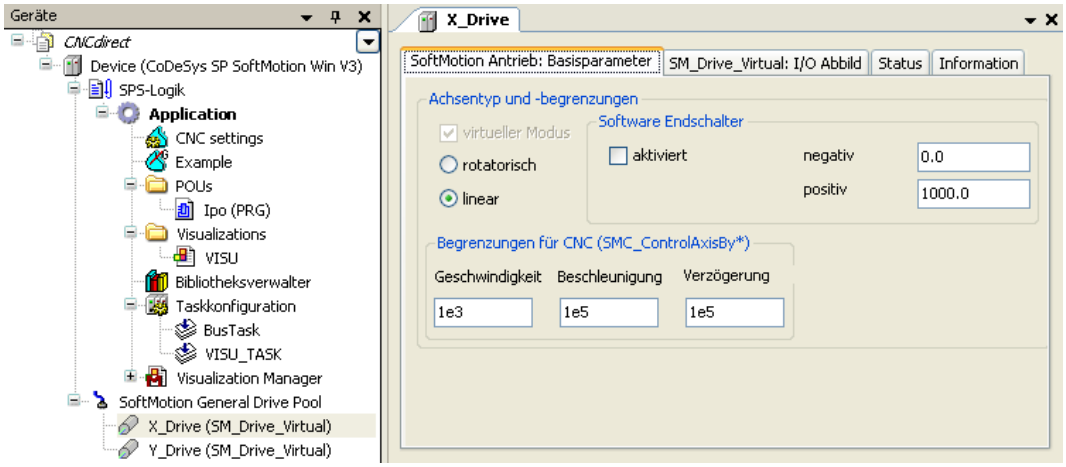


Abb. 525 Gerätebaum von CNCdirect

3. Erstellen des IEC-Programms:

Wir müssen zunächst die Antriebe mit dem MC_Power-Baustein aktivieren:

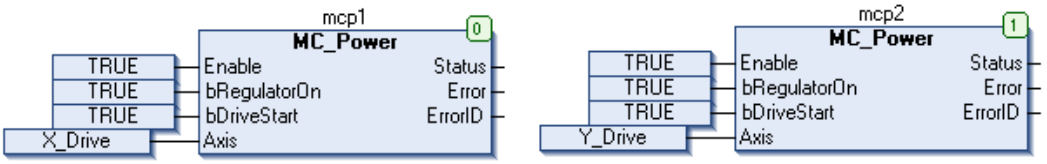


Abb. 526 MCpower Funktionsblock des Ipo Objekts

Als weiteres wichtiges Element benötigen wir einen SMC_Interpolator-Baustein. Dieser erhält als poqDataIn-Eingang die Adresse des erstellten CNC-Programms. Zudem muss die IEC-Task-Zykluszeit in dwIpoTime geschrieben werden. Diese kann man entweder als konstanten Wert dem Eingang dwIpoTime eingeben, oder man benützt die Variable dwCycle der Achsgruppen-Struktur aus der Steuerungskonfiguration, was den Vorteil hat, dass, wenn man die Zykluszeit der Task ändert, automatisch die richtige Zeit als Interpolator-Eingang verwendet wird.

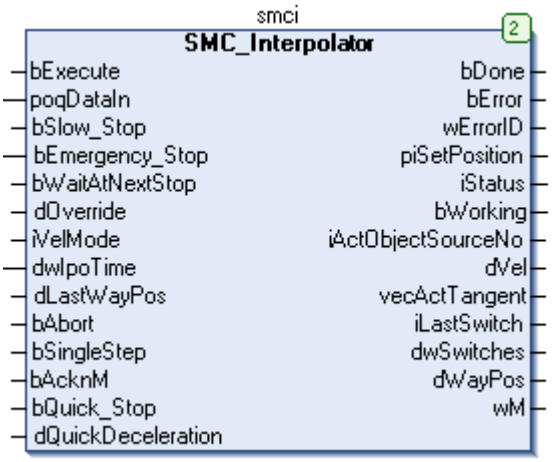


Abb. 527 Funktionsblock SMC_Interpolator des Ipo Objekts

In unserem Beispiel wollen wir ein Portal-System steuern. Deshalb fügen wir eine Instanz der Rückwärts- und Vorwärtstransformations-Bausteine aus der Bibliothek SM_Trafo.lib ein. Der Vorwärtstransformations-Baustein erhält als Eingänge die Antriebe (der Z-Antrieb wird mit einer sonst nicht verwendeten Variable dummy vom Typ AXIS_REF belegt); der Rückwärtstrafo-Baustein muss die Soll-Position des Interpolators erhalten.

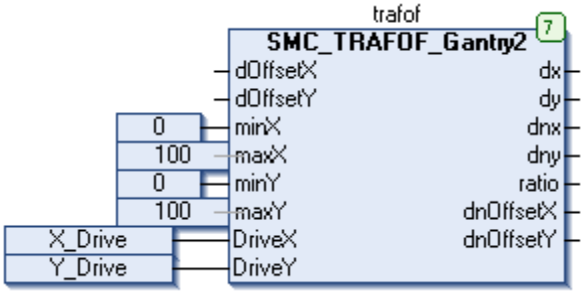


Abb. 528 Modul zur Vorwärtstransformation

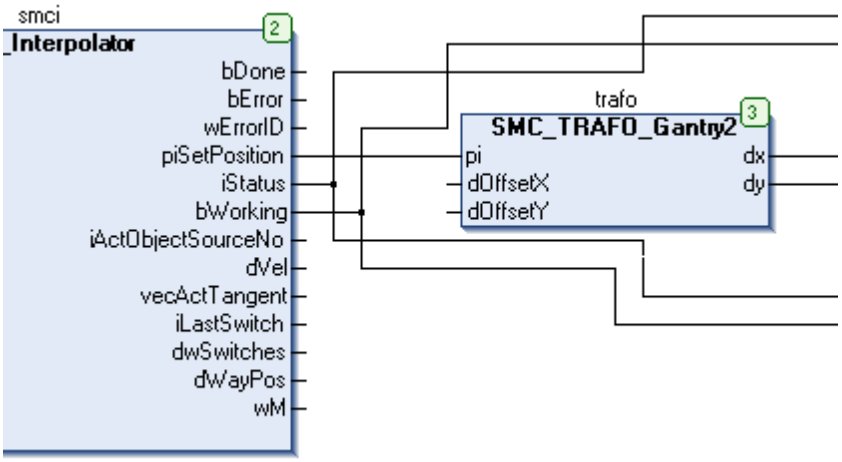


Abb. 529 Modul zur Rüchwärtstransformation

Die Ausgänge des Bausteins, also die Achs-Koordinaten, müssen nun auf die Antriebe geschrieben werden. Dafür existieren die Funktionsblöcke SMC_ControlAxisByPos. Da unsere Applikation nicht gewährleistet, dass die Ausgaben des Interpolators stetig sind, (zum Beispiel endet die Bahn an einem anderen Punkt als sie anfängt), sollten wir die Sprungvermeidung aktivieren (bAvoidGaps, fGapVelocity, fGapAcceleration, fGapDeceleration), den StopIpo-Ausgang mit dem bEmergency_Stop des Interpolators verbinden und den Interpolator-Ausgang iStatus mit den entsprechenden Eingängen der Achskontroll-Bausteine verbinden.

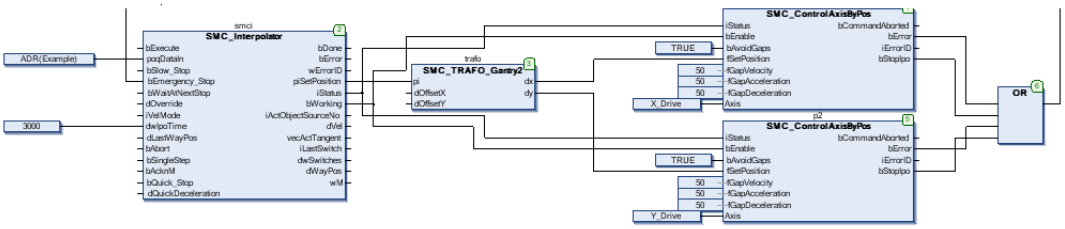


Abb. 530 Anbindung der verwendeten Funktionsblöcke des Ipo Objekts

Man beachte bei der Programmierung mit CFC vor allem auch auf die richtige Reihenfolge der Bausteine!

4. Erstellen der Bedien- und Testoberfläche:

Wir binden in eine neue Visualisierung zwei Visualisierungsobjekte ein. Zum einen das Template des Interpolators, zum anderen das Template der Transformation. Diese werden über das Platzhalterkonzept mit den entsprechenden Baustein-Instanzen (hier: Ipo.smci bzw. Ipo.trafof) verknüpft.

5. Inbetriebnahme:

Das so erstellte Programm lässt sich fehlerfrei übersetzen und starten und führt das CNC-Programm aus, sobald der Execute-Eingang des Interpolators gesetzt wurde. Wurde es vollständig abgefahren, kann durch eine neue steigende Flanke ein erneutes Abfahren erreicht werden.

Man beachte auch die Funktion der Wegschalter, die in der Visualisierung des Interpolations-Bausteins ebenfalls dargestellt sind.

CNC-Beispiel 2: Dekodierung online mit Verwendung von Variablen

Beachten Sie das dem Beispiel entsprechende Projekt "CNConline.project", das im Lieferumfang enthalten ist.

1. Erstellen des NC-Programms im CNC-Editor:

Wir erstellen wie im vorigen Beispiel ein CNC-Programm und verwenden darin zwei globale Variablen g_x und g_y. Z.B.:

Example:

```
VAR_GLOBAL
g_x: REAL:=100;
g_y:REAL:=50;
END_VAR
```

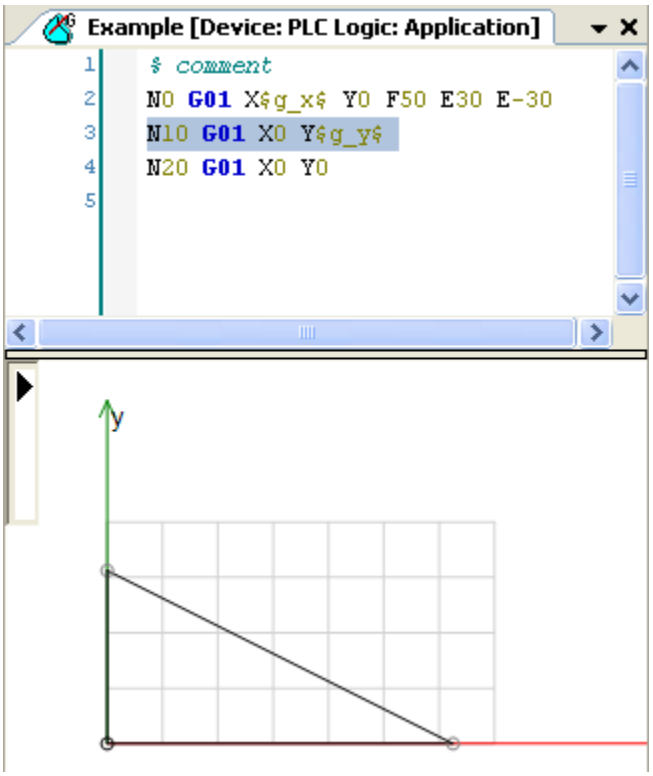


Abb. 531 Example Objekt

Als Übersetzungsmodus wählen wir diesmal SMC_CNC_REF, da wir Variablen in unserem Programm verwenden.

2. Drive Interface, Steuerungskonfiguration:

Die Antriebsstruktur entspricht der des vorhergehenden Beispiels.

3. Erstellen des IEC-Programms:

Da wir diesmal den anderen Übersetzungsmodus gewählt haben, müssen wir das Decodieren und Bahnvorverarbeiten im IEC-Programm durchführen. Da dieser eher laufzeitintensive Vorgang nicht im Interpolator-Takt durchgeführt werden muss (es wird ja pro Decoder-Aufruf ein Wegobjekt erzeugt, welches typischerweise für viele Interpolator-Aufrufe reicht), wird er häufig in eine weniger gewichtete und seltener aufgerufene Task ausgelagert.

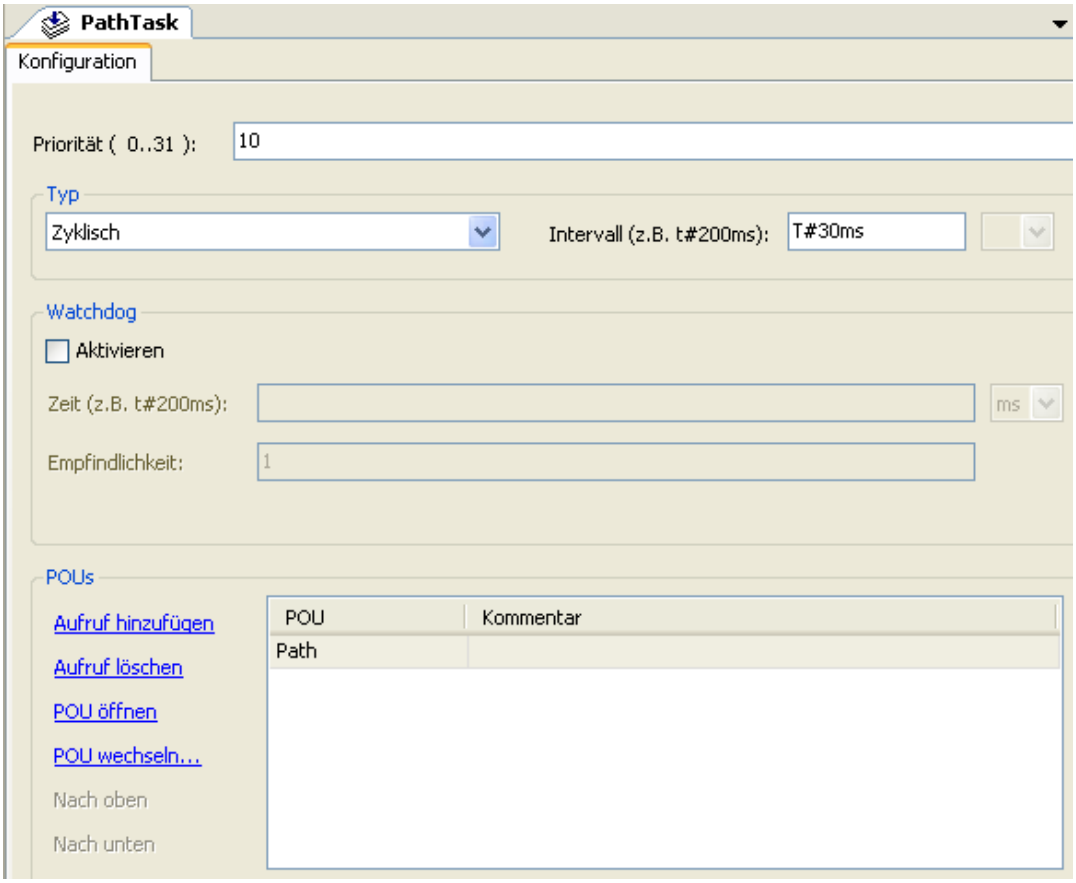


Abb. 532 PathTask Objekt

Der zugrunde liegende Mechanismus ist folgender: In der langsamen Task wird anfänglich pro Zyklus etwa ein GEOINFO-Objekt erzeugt, welches in der OUTQUEUE-Struktur des Decoder-Bausteins gespeichert ist. Ist diese OUTQUEUE voll, pausieren die Bausteine der langsamen Task solange, bis die OUTQUEUE nicht mehr voll ist, sprich bis die schnelle Task das erste GEOINFO-Objekt abgearbeitet hat und dieses aus der OUTQUEUE löscht. Dann werden die Bausteine der langsamen Task wieder aktiv und befüllen die OUTQUEUE-Struktur. In der schnellen Task wird pro Zyklus ein Bahnpunkt aus der OUTQUEUE-Struktur, auf die der Eingang DataIn des Interpolators zeigt, berechnet und verarbeitet. Da ein GEOINFO-Objekt ja in der Regel aus mehreren Bahnpunkten besteht, dauert es einige Zyklen bis das erste GEOINFO-Objekt abgearbeitet ist und vom Interpolator automatisch gelöscht wird. Da das Abarbeiten eines GEOINFO-Objekts offenbar mehrerer Zyklen bedarf als das Erstellen, kann die langsame Task tatsächlich seltener aufgerufen werden als die schnelle. Die Task-Zeiten müssen jedoch so gewählt werden, dass in der letzten OUTQUEUE der langsamen Task immer genügend GEOINFO-Objekte gelagert sind, sodass kein Data-Underrun auftritt. Dies tritt auf, wenn dem Interpolator aus DataIn keine GEOINFO-Objekte mehr zur Verfügung stehen, und das Bahnende noch nicht erreicht ist. In diesem Fall bremst der Interpolator und bleibt so lange stehen, bis wieder neue Datenelemente verfügbar sind.

Im Programm Path findet die Decodierung des NC-Programms zur OUTQUEUE und die Geschwindigkeitsüberprüfung statt:

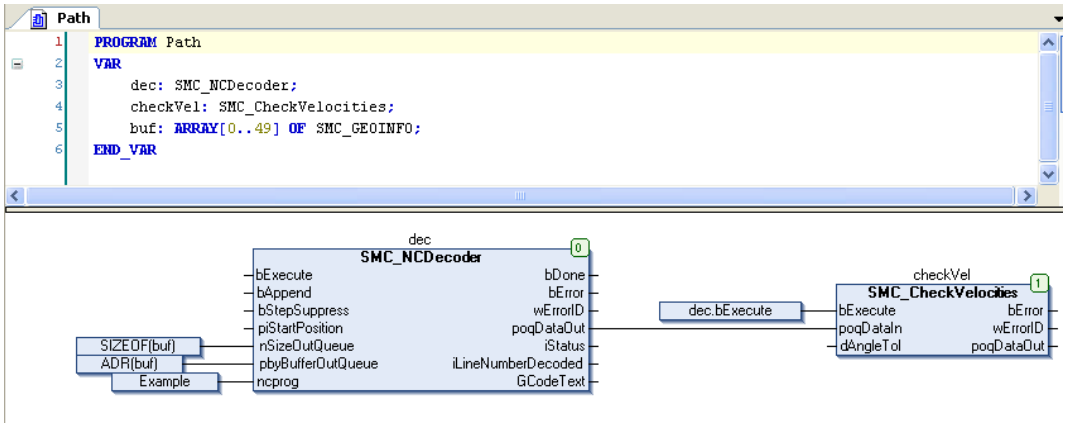


Abb. 533 Path Objekt

Der interpolierende Teil des IEC-Programm bleibt fast gleich, nur dass der Daten-Eingang des Interpolators nicht wie zuvor dem CNC-Programmnamen (ADR(Example)) entspricht, sondern aus dem OutQueue-Ausgang der bahnvorverarbeitenden Bausteine (g_CheckVel.poqDataOut) besteht.

4. Erstellen der Bedien- und Testoberfläche:

Zu der Visualisierung des vorigen Beispiels ist es sinnvoll, Templates der neuen Bausteine (SMC_NCDecoder und SMC_CheckVelocities) hinzuzufügen. Außerdem sollte man die globalen Variablen g_x und g_y ändern können, um später bei der Inbetriebnahme deren Funktion überprüfen zu können.

5. Inbetriebnahme:

Das so erstellte Programm lässt sich fehlerfrei übersetzen und starten und führt das CNC-Programm aus, nachdem die Execute-Eingänge des Decoders und Interpolators gesetzt wurden. Ändert man die Werte der globalen Variablen, werden diese bei einem Neustart des Decoders übernommen und die Bahn passt sich entsprechend an. Man beachte auch die Funktion des Append-Eingangs des Decoders.

CNC-Beispiel 3: Bahnvorverarbeitung online

Beachten Sie das dem Beispiel entsprechende Projekt "CNCprepro.project", das im Lieferumfang enthalten ist.

Wir wollen Beispiel 2 um eine Bahnvorverarbeitung erweitern: Die Ecken des letzten Programms sollen durch Splines verrundet werden. Dies führt der Baustein SMC_SmoothPath durch. Das CNC-Programm muss von den Worten G51/G50 umklammert werden, dann ergibt sich:

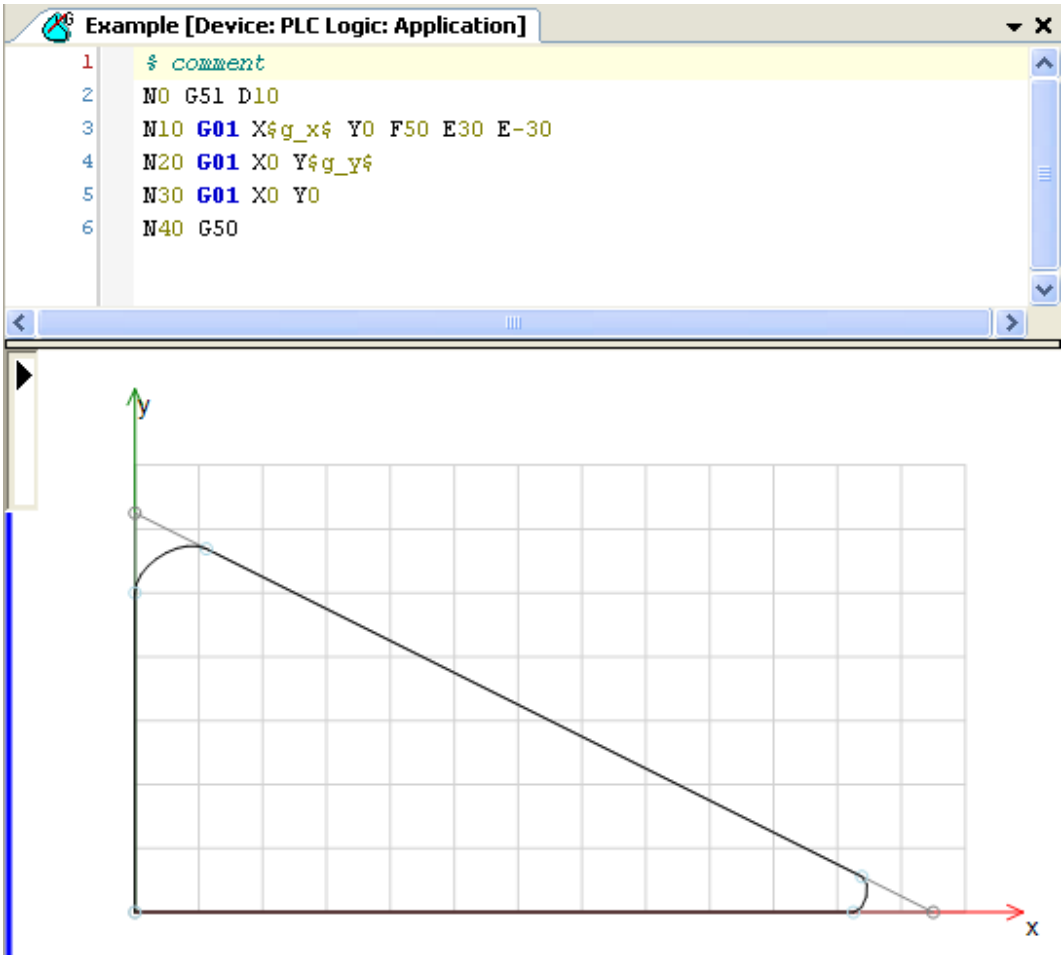


Abb. 534 Example Object des "CNCprepro.project"s

Würden wir keine Variablen verwenden, könnten wir das Programm in dieser Form als Queue übersetzen und direkt in den Interpolator eingeben; da wir aber Variablen verwenden, müssen wir das Decodieren und die Eckverschleifung selbst durchführen. Deshalb deklarieren wir einen neuen Baustein vom Typ SMC_SmoothPath und rufen ihn nach dem Decoder auf.

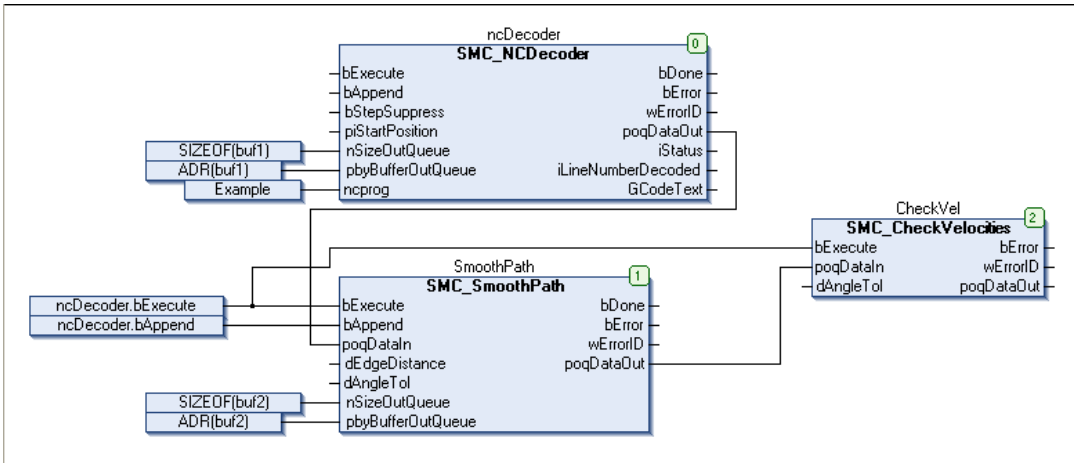


Abb. 535 Path Objekt des "CNCprepro.project"s

Der Dateneingang des Interpolator-Bausteins muss wie immer auf den Ausgang **poqDataOut** des CheckVelocities-Bausteins gesetzt werden.

Dieses Programm kann fehlerfrei übersetzt werden und stoppt – im Unterschied zum vorigen – nicht mehr in den Ecken des NC-Programms, da die Ecken der Bahn durch die Bahnvorverarbeitung knickfrei gemacht wurde.

CNC-Beispiel 4: CNC mittels Tabelleneditor

Beachten Sie das dem Beispiel entsprechende Projekt **CNC_table.project**, das im Lieferumfang enthalten ist.

Hier ist die Funktionalität des Projekts **CNCprepro.project** mit dem Tabelleneditor des CNC-Editors programmiert. Im Unterschied zu **CNCprepro** werden in **CNC_table** keine IEC-Variablen verwendet, sondern diese mit einem festen Werten belegt. Ansonsten ist die Funktionalität identisch.

CNC-Beispiel 5: CNC aus Datei

Beachten Sie das dem Beispiel entsprechende Projekt **CNC_File.project**, das im Lieferumfang enthalten ist.

Anhand dieses Beispiels kann nachvollzogen werden, wie das Einlesen einer ASCII-Datei mit G-Code Programm im IEC-Code funktioniert.

Dies wird dann verwendet, wenn das G-Code Programm sehr groß ist.

Hier ist die Funktionalität des Projekts ähnlich der in **CNCOnline.project**. Das G-Code-Programm wird aber aus dem ASCII-File eingelesen. Auch werden keine Variablen verwendet, sondern feste Werte gesetzt und die Abarbeitungsreihenfolge ist vergleichbar der in **CNC_Online.project**.

Zusammenfassend lassen sich folgende Unterschiede feststellen:

1. zusätzlicher virtueller Antrieb (z-Achse)
2. Baustein **smoothpath** wird nicht benötigt
3. G-Code wird aus Datei eingelesen
4. Vorverarbeitung online (siehe **CNC-Online.project**)
5. zusätzlich Baustein ReadNCFile
6. Visu-Template vorhanden.

15 Kommandozeilen-Optionen

15.1 Starten von PLC Designer.exe

Die folgende Kommandozeilen-Option kann beim Start von PLC Designer.exe von der Kommandozeile aus angegeben werden.

- --culture

15.1.1 --culture

Diese Option kann beim Starten von PLC Designer.exe über Kommandozeile hinzugefügt werden, um die Benutzeroberfläche in einer der verfügbaren Landessprachen zu erhalten.

Syntax:

--culture=<Culture>

wobei <Culture> das übliche Sprachkürzel für die gewünschte Landessprache ist, beispielsweise "de", "en", "fr", "it", "es", "zh-CHS", etc.

Beispiel:

PLC Designer.exe --culture=en

zum Starten der englischen Benutzeroberfläche.

16 Programmierreferenz

16.1 Deklarieren

Die Variablen eines Projekts müssen entweder manuell im Deklarationseditor oder mit Hilfe des Autodeklarations-Dialogs deklariert werden. Sehen Sie dazu die entsprechenden Hilfeseiten zu den verschiedenen Variablen-Kategorien (lokal, global, input, output etc.), zur Initialisierung, zum Gebrauch von Pragmas, Init Methoden etc.

16.1.1 Variablendeklaration

Die Deklaration einer Variablen kann im Deklarationsteil einer POU , im Autodeklarations-Dialog wie auch in einem DUT- oder GVL-Editor vorgenommen werden.

Der Typ (im Autodeklarationsdialog als "Sichtbarkeit " bezeichnet) der zu deklarierenden Variablen wird durch die Schlüsselwörter angegeben, die die Deklaration "einklammern". Die Deklaration einfacher Variablen beispielsweise beginnt mit "VAR" und wird mit "END_VAR" abgeschlossen. Weiter Typen:

- VAR_INPUT
- VAR_OUTPUT
- VAR_IN_OUT
- VAR_GLOBAL
- VAR_TEMP
- VAR_STAT
- VAR_EXTERNAL
- VAR_CONFIG

Die Variablentypen-Schlüsselwörter können durch Attribut-Schlüsselwörter ergänzt werden. Beispiel: "RETAIN" (VAR_INPUT RETAIN).

Eine Variablendeklaration muss den folgenden Regeln folgen:

Syntax:

<Bezeichner> {AT <Adresse>}:<Datentyp> {:=<Initialisierung>};

Die Teile in geschweiften Klammern {} sind optional.

Bezeichner: Der Bezeichner ist der Name der Variable. Die im folgenden aufgelisteten Punkte müssen bei der Vergabe eines Bezeichner zwingend beachtet werden, aber es gibt auch zusätzliche Empfehlungen.

- Ein Bezeichner darf keine Leerzeichen oder Sonderzeichen enthalten.
- Groß-/Kleinschreibung bei Bezeichner wird nicht beachtet, d.h. "VAR1", "Var1" und "var1" bezeichnen dieselbe Variable.
- Der Unterstrich wird erkannt (d.h. zum Beispiel "A_BCD" und "AB_CD" werden als zwei verschiedene Bezeichner behandelt), aber mehrfach aufeinanderfolgende Unterstriche sind nicht erlaubt.
- Die Länge eines Bezeichners wie auch des signifikanten Teils sind unbegrenzt.
- Die in der folgenden Textbox beschriebenen Regeln bezüglich der Mehrfachverwendung von Kennzeichnern müssen beachtet werden:

Mehrfachverwendung von Bezeichnern (Namensräume):

- Ein Bezeichner darf lokal nicht doppelt verwendet werden.
- Ein Bezeichner darf nicht identisch mit einem Schlüsselwort sein.
- Global darf ein Bezeichner mehrfach verwendet werden, also kann eine lokale Variable den selben Namen haben wie eine globale. In diesem Fall hat innerhalb einer POU die lokale Variable Vorrang.
- Eine Variable, die in einer Globalen Variablenliste definiert ist, kann denselben Namen haben wie eine Variable, die in einer anderen GVL definiert ist. In diesem Zusammenhang beachten Sie bitte die folgenden norm-erweiternden Funktionalitäten bezüglich des **Namensraums**/Gültigkeitsbereichs von Variablen, die in »PLC Designer« V2.3 noch nicht verfügbar waren:

1. "Global scope" Operator: Ein Instanzpfad, der mit "." beginnt, öffnet immer einen globalen Namensraum. Wenn es eine lokale Variable, z.B. "ivar", gibt, die denselben Namen hat wie eine globale Variable, wird mit ".ivar" die globale angesprochen.
2. Der Name einer Globalen Variablenliste kann den Namensraum für die enthaltenen Variablen eindeutig definieren. Somit können Variablen mit demselben Namen in verschiedenen Globalen Variablenlisten deklariert werden und dennoch durch das Voranstellen des Listennamens eindeutig angesprochen werden.
   Beispiel:
   globlist1.ivar := globlist2.ivar; (* ivar aus GVL globlist2 wird auf ivar in GVL globlist1 kopiert *)
3. Variablen, die in der Globalen Variablenliste einer ins Projekt eingebundenen Bibliothek definiert sind, können gemäß der Syntax "<Namensraum Bibliothek>.<Name der GVL>.<Variablenname>" eindeutig angesprochen werden. Sehen Sie im folgenden Punkt weiteres zum Namensraum von Bibliotheken.
   Beispiel:
   globlist1.ivar := lib1.globlist1.ivar (* ivar aus GVL globlist1 in Bibliothek lib1 wird auf ivar in GVL globlist1 kopiert *)

- Für eine Bibliothek wird beim Einfügen via Bibliotheksverwalter auch ein Namensraum definiert. Somit kann ein Bibliotheksbaustein bzw. eine Bibliotheksvariable über "<Namensraum Bibliothek>.<Bausteinname|Variablename>" eindeutig angesprochen werden. Beachten Sie im Fall von verschachtelten Bibliotheken, dass die Namensräume aller beteiligten Bibliothek in Folge anzugeben sind.
  Beispiel: Wenn Lib1 durch Lib0 referenziert wird, so wird der in Lib1 enthaltene Baustein fun über
  "Lib0.Lib1.fun" angesprochen:
  ivar := Lib0.Lib1.fun(4, 5); (* Rückgabewert von fun wird auf Variable ivar im Projekt kopiert *)
  Sofern in den Eigenschaften der referenzierten Bibliothek Lib1 das Kontrollkästchen "Alle IEC-Symbole ins referenzierende Projekt veröffentlichen." aktiviert ist, kann fun auch direkt über "Lib0.fun" angesprochen werden.

AT <Adresse>: Die Variable kann mit Hilfe des Schlüsselwortes "AT" direkt an eine bestimmte Adresse gebunden werden.

In Funktionsblöcken können Sie auch Variablen mit unvollständigen Adreßangaben deklarieren. Damit eine solche Variable in einer lokalen Instanz verwendet werden kann, muss allerdings ein entsprechender Eintrag für diese Variable in der "Variablenkonfiguration" vorliegen.

Datentyp: gültiger Datentyp, optional erweitert durch eine ":=<Initialisierung>".

Optional können im Deklarationsteil eines Objekts Pragma-Anweisungen hinzugefügt werden, um die Code-Generierung für verschiedene Zwecke zu beeinflussen.

Hinweise:
Beachten Sie die Möglichkeit der automatischen Deklaration.
Für eine schnellere Eingabe von Deklarationen kann der Kurzform-Modus verwendet werden.

16.1.2 Empfehlungen für die Vergabe von Bezeichnern

Bezeichner werden bei der Deklaration von Variablen (Variablennamen),benutzerdefinierten Datentypen und beim Anlegen von POUs (Bausteine: Funktionen, Funktionsblöcke, Programme) und Visualisierungen vergeben. Zusätzlich zu den Punkten, die sowieso beachtet werden müssen, wird empfohlen die folgenden Regeln anzuwenden, um eine möglichst weitgehende Vereinheitlichung bei der Namensvergabe zu erreichen:

Variablennamen

Variablennamen in »PLC Designer« V3.x Bibliotheken

Benutzerdefinierte Datentypen (DUT)

Benutzerdefinierte Datentypen (DUTs) in »PLC Designer« V3.x Bibliotheken

Funktionen, Funktionsblöcke, Programme (POU), Aktionen

POUs in »PLC Designer« V3.x Bibliotheken

Bezeichner für Visualisierungen

(1) Variablennamen

Die Benennung von Variablen soll in allen Applikationen und Bibliotheken möglichst angelehnt an die **ungarische Notation** erfolgen:

Für jede Variable sollte eine sinnvolle, möglichst kurze, englische Beschreibung gefunden werden, der **Basisname**. Der jeweils erste Buchstabe eines Wortes des Basisnamens soll groß, die übrigen klein geschrieben werden (Bsp.: FileSize). Bei Bedarf kann eine Übersetzungsdatei in andere Sprachen zusätzlich erstellt werden.

Vor diesen Basisnamen werden, entsprechend des Datentyps der Variablen, **Präfix(e)** in Kleinbuchstaben gehängt.

Datentyp	Untergrenze	Obergrenze	Informations-gehalt	Präfix	Kommentar
BOOL	FALSE	TRUE	1 Bit	x*	
				b	reserviert
BYTE			8 Bit	by	Bitstring, nicht für arithm. Operationen
WORD			16 Bit	w	Bitstring, nicht für arithm. Operationens
DWORD			32 Bit	dw	Bitstring, nicht für arithm. Operationen
LWORD			64 Bit	lw	nicht für arithm. Operationen
SINT	-128	127	8 Bit	si	
USINT	0	255	8 Bit	usi	
INT	-32.768	32.767	16 Bit	i	

Datentyp	Untergrenze	Obergrenze	Informations-gehalt	Präfix	Kommentar
UINT	0	65.535	16 Bit	ui	
DINT	-2.147.483.648	2.147.483.647	32 Bit	di	
UDINT	0	4.294.967.295	32 Bit	udi	
LINT	-2^{63}	$2^{63} - 1$	64 Bit	li	
ULINT	0	$2^{64} - 1$	64 Bit	uli	
REAL			32 Bit	r	
LREAL			64 Bit	lr	
STRING				s	
TIME				tim	
TIME_OF_DAY				tod	
DATE_AND_TIME				dt	
DATE				date	
ENUM			16 Bit	e	
POINTER				p	
ARRAY				a	

* bewußt wurde für BOOLsche Variablen x als Präfix gewählt, um zum einen eine Abgrenzung zu BYTE zu finden, und zum anderen um der Sicht des IEC-Programmierers (vgl. Adressierung %IX0.0) entgegenzukommen.

Beispiele:

```
bySubIndex: BYTE;
sFileName: STRING;
udiCounter: UDINT;
```

Bei **verschachtelten Deklarationen** werden die Präfixe in der Reihenfolge der Deklaration aneinandergehängt:

Beispiel:

```
pabyTelegramData: POINTER TO ARRAY [0..7] OF BYTE;
```

Funktionsblock-Instanzen und **Variablen** benutzerdefinierter Datentypen erhalten als Präfix einen Kurzbezeichner für den FB- bzw. Datentypnamen (Bsp.: sdo).

Beispiel:

```
cansdoReceivedTelegram: CAN_SDOTelegram;

TYPE CAN_SDOTelegram :    (* prefix: sdo *)
STRUCT
  wIndex:WORD;
  bySubIndex:BYTE;
  byLen:BYTE;
  aby: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE
```

Lokale Konstanten (c) beginnen mit dem Konstanten-Präfix c und einem zusätzlichen Unterstrich, gefolgt vom Typ-Präfix und dem Variablennamen.

Beispiel:

```
VAR CONSTANT
  c_uiSyncID: UINT := 16#80;
END_VAR
```

Für Globale Variablen (g) und Globale Konstanten (gc) wird an das Bibliothekspräfix ein zusätzliches Präfix + Unterstrich angehängt:

Beispiele:

```
VAR_GLOBAL
  CAN_g_iTest: INT;
END_VAR
VAR_GLOBAL CONSTANT
  CAN_gc_dwExample: DWORD;
END_VAR
```

(2) Variablennamen in »PLC Designer« V3.x Bibliotheken

Die Variablennamen-Bildung in »PLC Designer« V3.x entspricht obiger Beschreibung mit der Ausnahme, daß globale Variablen und Konstanten kein Bibliothekspräfix benötigen, da dessen Funktion durch den Namespace ersetzt wird.

Beispiel:

g_iTest: INT; (Deklaration)

CAN.g_iTest (Verwendung, Aufruf im Programm)

(3) Benutzerdefinierte Datentypen (DUT)

Strukturen: Der Name jedes Struktur-Datentypen besteht aus dem Bibliothekspräfix (Bsp.: CAN), einem Unterstrich und einer kurzen, aussagekräftigen Beschreibung (Bsp.: SDOTelegram) der Struktur. Das zugehörige Präfix für angelegte Variablen dieser Struktur soll als Kommentar direkt nach dem Doppelpunkt folgen.

Beispiel:

```
TYPE CAN_SDOTelegram :    (* prefix: sdo *)
STRUCT
  wIndex:WORD;
  bySubIndex:BYTE;
  byLen:BYTE;
  abyData: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE
```

Enumerationswerte beginnen mit dem Bibliothekspräfix (Bsp.: CAL), gefolgt von einem Unterstrich und dem Bezeichner in Großbuchstaben.

Man beachte, dass in vergangenen »PLC Designer«-Versionen ENUM-Werte > 16#7FFF zu Fehlern geführt haben, da sie nicht automatisch in INT konvertiert wurden. Aus diesem Grund sollen ENUMs stets mit korrekten INT-Werten definiert werden.

Beispiel:

```
TYPE CAL_Day :(
CAL_MONDAY,
CAL_TUESDAY,
CAL_WEDNESDAY,
CAL_THIRSDAY,
CAL_FRIDAY,
CAL_SATURDAY,
CAL_SUNDAY);
```

Deklaration:

```
eToday: CAL_Day;
```

(4) Benutzerdefinierte Datentypen (DUTs) in »PLC Designer« V3.x Bibliotheken

Bei DUT-Namen wird in »PLC Designer« V3.x Bibliotheken auf das Bibliothekspräfix verzichtet, da dessen Funktion durch den Namespace ersetzt wird. Ebenfalls werden die Enumerationswerte ohne Bibliothekspräfix definiert:

Beispiel (aus Bibliothek mit Namespace CAL):

```
TYPE Day :(
MONDAY,
TUESDAY,
WEDNESDAY,
THIRSDAY,
FRIDAY,
SATURDAY,
SUNDAY);
```

Deklaration:

```
eToday: CAL.Day;
```

Verwendung in der Applikation:

```
IF eToday = CAL.Day.MONDAY THEN
```

(5) Funktionen, Funktionsblöcke, Programme (POU)

Funktionen, Funktionsblöcke und Programme bestehen aus dem Bibliothekspräfix (Bsp.: CAN), einem Unterstrich und einem aussagekräftigen, möglichst kurzen Namen der POU (Bsp.: SendTelegram). Wie bei den Variablen soll der jeweils erste Buchstabe eines Wortes des Basisnamens groß, die übrigen klein geschrieben werden. Es wird empfohlen, den Namen der POU aus einem Verb und einem Substantiv zusammenzusetzen.

Beispiel:

```
FUNCTION_BLOCK CAN_SendTelegram (* prefix: canst *)
```

Im Deklarationsteil sei als Kommentar eine Kurzbeschreibung des Bausteins enthalten. Außerdem werden alle Ein- und Ausgänge mit Kommentaren versehen. Im Fall von Funktionsblöcken soll das zugehörige Präfix für angelegte Instanzen als Kommentar direkt nach dem Namen folgen.

Aktionen erhalten kein Präfix; nur Aktionen, die rein intern, also vom Baustein selbst, aufgerufen werden sollen, beginnen mit prv_.

Jede Funktion muss aus Kompatibilitätsgründen zu früheren »PLC Designer«-Versionen mindestens einen Übergabeparameter haben. Externe Funktionen dürfen keine Strukturen als Rückgabewert verwenden.

(6) POUs in »PLC Designer« V3.x Bibliotheken

Bei POU-Namen wird in »PLC Designer« V3.x Bibliotheken auf das Bibliothekspräfix verzichtet, da dessen Funktion durch den Namespace ersetzt wird. Für die Namensbildung von Methoden gelten dieselben Regeln wie für Aktionen. Mögliche Eingänge der Methoden sollen mit englischsprachigen Kommentaren versehen werden. Ebenso soll eine Kurzbeschreibung der Methode in deren Deklaration enthalten sein. Für Methoden gibt es keine Einschränkungen bezüglich der Implementierbarkeit zwischen externen und internen Bibliotheken. Interfaces sollen mit dem Buchstaben I beginnen, z.B.: ICANDevice

(7) Bezeichner für Visualisierungen

Hinweis: Momentan muss darauf geachtet werden, dass eine Visualisierung nicht denselben Namen erhält wie ein anderer Baustein im Projekt, da dies bei Visualisierungswechseln zu Problemen führen würde.

16.1.3 Variablen-Initialisierung

Der **Standard-Initialisierungswert** für alle Deklarationen ist "0", aber es können im Deklarationsteil jeder Variable und jedes Datentyps auch benutzerdefinierte Initialisierungswerte angegeben werden.

Die **benutzerdefinierte Initialisierung** wird mit dem Zuweisungsoperator "**:=**" begonnen und kann aus einem beliebigen, gültigen ST-Ausdruck bestehen. Somit kann der Initialwert mit Hilfe von Konstanten, anderen Variablen oder Funktionen definiert werden. Der Programmierer muss nur sicherstellen, dass eine für die Initialisierung einer anderen Variable verwendete Variable selbst ebenfalls deklariert ist.

Beispiele für gültige Variablen-Initialisierungen:

```
VAR
var1:INT := 12;          (* Integer-Variable mit Initialwert 12*)
x : INT := 13 + 8;       (* Initialwert durch einen Ausdruck von Konstanten definiert*)
y : INT := x + fun(4);   (* Initialwert durch einen Ausdruck definiert, der einen Funktionsaufruf enthält; beachten Sie in solchen Fällen die Reihenfolge! *)
z : POINTER TO INT := ADR(y);   (* Nicht in der Norm IEC61131-3 beschrieben: Initialwert durch eine Adressfunktion definiert; Bedenken Sie allerdings in diesem Fall, dass der Zeiger während eines Online Changes nicht initialisiert wird! *)
END_VAR
```

Siehe auch: Beliebige Ausdrücke für Variablen-Initialisierung

Hinweis: Ab Compiler-Version 3.3.2.0 werden Variablen aus Globalen Variablenlisten immer vor den lokalen Variablen einer POU initialisiert.

16.1.4 Beliebige Ausdrücke für Variablen-Initialisierung

Eine Variable kann mit einem beliebigen ST-Ausdruck initialisiert werden. Es können andere Variablen aus dem gleichen Namensraum wie auch Funktionsaufrufe verwendet werden. Jedoch muss sichergestellt sein, dass die für die Initialisierung einer anderen Variable verwendeten Variablen selbst ebenfalls initialisiert werden.

Beispiele für gültige Variablen-Initialisierungen:

```
VAR
x : INT := 13 + 8;
y : INT := x + fun(4);
z : POINTER TO INT := ADR(y); (* Achtung: Der Zeiger wird im Falle eines Online Change nicht initialisiert! *)
END_VAR
```

16.1.5 Deklarationseditor

Der Deklarationseditor ist ein Text- oder Tabelleneditor für die Deklaration von Variablen. Üblicherweise wird er in Verbindung mit den Spracheditoren verwendet.

Siehe Deklarationseditor für weitere Informationen.

16.1.6 Autodeklarations-Funktion

In den Optionen, Kategorie 'Texteditor' / 'Editieren', kann eingestellt werden, dass automatisch der Dialog 'Variable deklarieren' öffnen soll, sobald eine noch nicht deklarierte Variable im Implementationsteil eines Editors eingegeben und die <Eingabe>-Taste gedrückt wird. Dieser Dialog unterstützt die Deklaration der Variable.

Der Dialog kann auch explizit über Befehl 'Variable deklarieren' geöffnet werden, der standardmäßig im Bearbeiten-Menü enthalten ist, oder mit Tastenkombination <Umschalt>+<F2>. Wenn vor dem Öffnen eine bereits deklarierte Variable selektiert wird, kann im Dialog deren Deklaration bearbeitet werden.

16.1.7 Kurzform-Modus

Im Deklarationseditor wie auch in anderen Texteditoren, in denen Variablendeklarationen möglich sind, wird der "Kurzform-Modus" der Eingabe unterstützt.

Dieser Modus wird aktiviert, wenn Sie eine Deklarationszeile mit der Tastenkombination **<Strg>+<Eingabe>** beenden, und er ermöglicht, Kurzformen für die Eingabe einer Deklaration zu verwenden anstatt sie ausführlich einzutippen.

Die folgenden Kurzformen werden unterstützt:

- Alle Bezeichner bis auf den letzten Bezeichner einer Zeile werden zu Variablenbezeichnern einer Deklaration.
- Der Datentyp der Deklaration wird durch den letzten Bezeichner der Zeile bestimmt. Hierbei gilt:

B or BOOL	ergibt	BOOL
I or INT	ergibt	INT
R or REAL	ergibt	REAL
S or string	ergibt	STRING

- Wenn durch diese Regeln kein Datentyp festgelegt werden konnte, wird automatisch BOOL als Datentyp verwendet und der letzte Bezeichner wird nicht als Datentyp benutzt (siehe unten: Beispiel 1).
- Jede eingegebene Konstante wird, je nach Typ der Deklaration, zu einer Initialisierung oder einer Stringlängenangabe (siehe unten, Beispiele 2 und 3).
- Eine Adresse (wie in %MD12) wird automatisch um das "AT"-Attribut erweitert (siehe unten, Beispiel 4).
- Ein Text nach einem Strichpunkt (;) wird zu einem Kommentar (siehe unten, Beispiel 4).
- Alle anderen Zeichen in der Zeile werden ignoriert (siehe unten, Ausrufezeichen in Beispiel 5).

Beispiele:

	Kurzform	resultierende Deklaration
(1)	A	A: BOOL;
(2)	A B I 2	A, B: INT := 2;
(3)	ST S 2; A string	ST:STRING(2); (* A string *)
(4)	X %MD12 R 5 Real Number	X AT %MD12: REAL := 5.0;(* Real Number *)
(5)	B !	B: BOOL;

16.1.8 AT-Deklaration

Es ist möglich, eine Projektvariable in der Deklaration an eine bestimmte Eingabe-, Ausgabe- oder Speicheradresse der im Gerätebaum konfigurierten Steuerung zu binden. Genauso kann die Zuweisung von Variablen auf eine Adresse im Mapping-Dialog der Gerätes in der Steuerungskonfiguration festgelegt werden.

Syntax:

<Bezeichner> AT <Adresse> : <Datentyp>;

Dem Schlüsselwort **AT** muss entsprechend der aktiven Steuerungskonfiguration im Gerätebaum eine **gültige Adresse** folgen. Siehe **Adresse** für weitere Informationen und beachte mögliche Speicherbereichsüberlappungen im Fall von Byte-Adressierungsmodus.

Diese Deklaration ermöglicht, der Adresse einen aussagekräftigen Namen zu geben. Eventuell nötige Änderungen bzgl. des ein- oder ausgehenden Signals dürfen nur an einer Stelle durchgeführt werden (z. B. in der Deklaration).

Beachten Sie das folgende beim Zuweisen einer Variable zu einer Adresse:

- Auf Variablen, die auf einen Eingang gelegt sind, kann nicht schreibend zugegriffen werden. Dies wird vom Compiler abgefangen.
- AT-Deklarationen können nur für lokale und globale Variablen durchgeführt werden, aber nicht für Ein- und Ausgabevariablen von Bausteinen.
- AT-Deklarationen dürfen nicht in Verbindung mit VAR RETAIN oder VAR PERSISTENT verwendet werden!
- Wenn AT-Deklarationen bei Struktur- oder Funktionsblock-Komponenten verwendet werden, werden alle Instanzen denselben Speicher verwenden, was der Verwendung von "statischen Variablen" in klassischen Programmiersprachen wie z.B. "C" entspricht.
- Das Speicher-Layout von Strukturen ist ebenfalls zielsystemabhängig.

Beispiele:

```
counter_heat7 AT %QX0.0: BOOL;
lightcabinetimpulse AT %IX7.2: BOOL;
download AT %MX2.2: BOOL;
```

Hinweis: Wenn boolsche Variablen einer Byte-, Word- oder DWORD-Adresse zugewiesen werden, belegen sie ein ganzes Byte mit TRUE oder FALSE, nicht nur das erste Bit nach dem Offset!

Beachten Sie, dass die Zuweisung einer Variable zu einer Adresse auch im Mapping-Dialog eines Geräts in der Steuerungskonfiguration (Geräte-Editor) erfolgen kann.

16.1.9 Schlüsselwörter

Schlüsselwörter müssen in allen Editoren in Großbuchstaben geschrieben werden.

Die folgenden Zeichenfolgen sind als Schlüsselwörter reserviert, d.h. sie können nicht als Bezeichner für Variablen oder POUs verwendet werden:

ABS
ACOS
ACTION (nur im Export-Format verwendet)
ADD
ADR
AND
ANDN
ARRAY
ASIN
AT
ATAN
BITADR
BOOL
BY
BYTE
CAL
CALC
CALCN
CASE
CONSTANT
COS
DATE
DINT
DIV
DO
DT
DWoRD
ELSE
ELSIF
END_ACTION (nur im Export-Format verwendet)
END_CASE
END_FOR
END_FUNCTION (nur im Export-Format verwendet)
END_FUNCTION_BLOCK (nur im Export-Format verwendet)

END_IF
END_PROGRAM (nur im Export-Format verwendet)
END_REPEAT
END_STRUCT
END_TYPE
END_VAR
END_WHILE
EQ
EXIT
EXP
EXPT
FALSE
FOR
FUNCTION
FUNCTION_BLOCK
GE
GT
IF
INDEXOF
INT
JMP
JMPC
JMPCN
LD
LDN
LE
LINT
LN
LOG
LREAL
LT
LTIME
LWORD
MAX
MIN
MOD
MOVE
MUL

MUX
NE
NOT
OF
OR
ORN
PARAMS
PERSISTENT
POINTER
PROGRAM
R
REFERENCE
READ_ONLY
READ_WRITE
REAL
REPEAT
RET
RETAIN
RETC
RETCN
RETURN
ROL
ROR
S
SEL
SHL
SHR
SIN
SINT
SIZEOF
SUPER
SQRT
ST
STN
STRING
STRUCT
SUPER
SUB

TAN
THEN
TIME
TO
TOD
TRUE
TRUNC
TYPE
UDINT
UINT
ULINT
UNTIL
USINT
VAR
VAR_ACCESS (nur sehr speziell und Hardware-abhängig verwendet)
VAR_CONFIG
VAR_EXTERNAL
VAR_GLOBAL
VAR_IN_OUT
VAR_INPUT
VAR_OUTPUT
VAR_STAT
VAR_TEMP
WHILE
WORD
WSTRING
XOR
XORN

Zusätzlich werden alle **Konvertierungs-Operatoren**, wie in der Eingabhilfe aufgelistet, als Schlüsselwörter behandelt.

16.1.10 Lokale Variablen - VAR

Zwischen den Schlüsselwörtern **VAR** und **END_VAR** werden alle lokalen Variablen eines Bausteins deklariert. Auf lokale Variablen kann von außen nicht geschrieben werden.

Beachten Sie die Möglichkeit, "VAR" durch ein Attribut zu erweitern.

Beispiel:

```
VAR
 iLoc1:INT; (* 1. Local Variable*)
END_VAR
```

16.1.11 Eingabevariablen - VAR_INPUT

Zwischen den Schlüsselwörtern **VAR_INPUT** und **END_VAR** werden Variablen deklariert, die als Eingabevariablen (Eingangsvariablen) eines Bausteines dienen. Das heißt, beim Aufruf des Bausteins kann diesen Variablen ein Wert übergeben werden.

Beachten Sie die Möglichkeit, "VAR_INPUT" mit einem Attribut zu erweitern.

Beispiel:

```
VAR_INPUT
 iIn1:INT (* 1. Inputvariable*)
END_VAR
```

16.1.12 Ausgabevariablen - VAR_OUTPUT

Zwischen den Schlüsselwörtern **VAR_OUTPUT** und **END_VAR** werden alle Ausgabevariablen (Ausgangsvariablen) des Bausteins deklariert. Das heißt, dass die Werte dieser Variablen an den aufrufenden Baustein zurückgeliefert werden. Sie können dort abgefragt und weiter verwendet werden.

Beachten Sie die Möglichkeit, "VAR_OUTPUT" mit einem Attribut zu erweitern.

Beispiel:

```
VAR_OUTPUT
 iOut1:INT; (* 1. Outputvariable*)
END_VAR
```

Ausgabevariablen in Funktionen und Methoden:

Gemäß der Norm IEC 61131-3 können Funktionen (und Methoden) zusätzliche Ausgänge haben. Diese müssen beim Aufruf der Funktion wie im folgenden Beispiel zugewiesen werden:

```
fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);
```

Der Rückgabewert der Funktion *fun* wird auf die - z.B. lokal deklarierten - Ausgangsvariablen (VAR_OUTPUT) *loc1* und *loc2* geschrieben.

16.1.13 EinAusgabevariablen - VAR_IN_OUT

Zwischen den Schlüsselwörtern **VAR_IN_OUT** und **END_VAR** werden alle Variablen deklariert die als Ein- und Ausgabevariablen des Bausteins dienen.

Hinweis: Bei EinAusgabevariablen wird direkt der Wert der übergebenen Variable verändert ("Übergabe als Pointer", Call-by-Reference). Das bedeutet, daß der Eingabewert für solche Variablen keine Konstante sein darf. Deshalb können auch die VAR_IN_OUT-Variablen eines Funktionsblocks nicht direkt von außen via <FBInstanz><EinAusgabevariable> gelesen bzw. beschrieben werden.

Beispiel:

```
VAR_IN_OUT
 iInOut1:INT; (* 1. Inputoutputvariable *)
END_VAR
```

16.1.14 Globale Variablen - VAR_GLOBAL

„Normale" Variablen, Konstanten, externe oder remanente Variablen, die im gesamten Projekt bekannt sein sollen, können als globale Variablen deklariert werden.

Hinweis: Eine lokal in einem Baustein deklarierte Variable, die denselben Namen hat wie eine globale Variable, hat innerhalb des Bausteins Vorrang.

Hinweis: Ab Compiler-Version 3.2.0.0 werden globale Variablen immer vor den lokalen Variablen von POUs initialisiert.

Die Variablen werden lokal zwischen den Schlüsselwörtern VAR_GLOBAL und END_VAR deklariert. Beachten Sie die Möglichkeit, "VAR_GLOBAL" mit einem Attribut zu erweitern.

Eine Variable wird als globale Variable erkannt, wenn dem Variablennamen ein Punkt vorangestellt wird, z.B. ".iGlobVar1".

Globale Variablenlisten dienen der Verwaltung von globalen Variablen in einem Projekt. Eine"GVL" kann über den Befehl Objekt hinzufügen als Objekt im Geräte- oder POUs-Fenster hinzugefügt werden.

16.1.15 Temporäre Variablen - VAR_TEMP

Diese Funktionalität ist eine **Erweiterung bezüglich der Norm IEC 61131-3**.

Temporäre Variablen werden bei jedem Aufruf des Bausteins neu initialisiert. VAR_TEMP Deklarationen sind nur in Programmen und Funktionsblöcken möglich. Auf die Variablen kann auch nur innerhalb des Implementationsteils eines Programms oder Funktionsblocks zugegriffen werden.

Die Variablen müssen lokal zwischen den Schlüsselwörtern VAR_TEMP und END_VAR deklariert werden.

16.1.16 Statische Variablen - VAR_STAT

Diese Funktionalität ist eine **Erweiterung bezüglich der Norm IEC 61131-3**.

Statische Variablen können in Funktionsblöcken, Funktionen und Methoden verwendet werden. Sie müssen lokal zwischen den Schlüsselwörtern VAR_STAT und END_VAR deklariert werden und werden beim ersten Aufruf des jeweiligen Bausteins initialisiert.

Auf statische Variablen kann nur innerhalb des Namensraums, in dem sie deklariert sind, zugegriffen werden (wie bei statischen Variablen in C), aber wie eine Globale Variable behalten sie ihren Wert auch nachdem der Baustein wieder verlassen wird. Sie können beispielsweise als Zähler für Funktionsaufrufe verwendet werden.

Beachten Sie die Möglichkeit, "VAR_STAT" mit einem Attribut zu erweitern.

16.1.17 Externe Variablen - VAR_EXTERNAL

Externe Variablen sind Globale Variablen die in einen Baustein "importiert" werden. Die Variablen müssen lokal zwischen den Schlüsselwörtern VAR_EXTERNAL and END_VAR deklariert werden. Die Deklaration muss der globalen exakt entsprechen. Wenn die globale Variable nicht existiert, wird eine Fehlermeldung ausgegeben.

Hinweis: Im »PLC Designer« ist es nicht notwendig, Variablen als extern zu deklarieren. Das Schlüsselwort ist vorgesehen, um die Kompatibilität zu IEC 61131-3 zu wahren.

Beispiel:

```
VAR_EXTERNAL
 iVarExt1:INT:=12; (* 1st external variable *)
END_VAR
```

16.1.18 Attribut-Schlüsselwörter für Variablentypen

Die folgenden Schlüsselwörter können in der Deklaration von Variablen dem Variablen"typ" hinzugefügt werden, um den Gültigkeits- (Sichtbarkeits-) bereich zu definieren:

RETAIN: Siehe Remanente Variablen vom Typ RETAIN

PERSISTENT: Siehe Remanente Variablen vom Typ PERSISTENT

CONSTANT: Siehe Konstanten

16.1.19 Remanente Variablen - RETAIN, PERSISTENT

Remanente Variablen können ihren Wert über die übliche Programmlaufzeit hinaus behalten. Sie werden als "Retain-Variablen" oder noch strenger als "Persistente Variablen" deklariert. Jeweils wird ein eigener Speicherbereich zur Verwaltung verwendet.

Die gewählte Art der Deklaration bestimmt den Grad der "Resistenz" einer remanenten Variablen im Falle von Resets, Downloads oder einem Reboot der Steuerung. In der Praxis wird vor allem die Kombination der beiden Typen gefordert sein (siehe unten Persistente Variablen).

Hinweis: Wenn ein V2.3-Projekt geöffnet wird, bleiben die Deklarationen von Retain-Variablen erhalten und unverändert wirksam, die Deklarationen von persistenten Variablen jedoch müssen überarbeiten bzw. neu erstellt werden, siehe unten: eine eigene globale Variablenliste muss erstellt werden!

Hinweis: Eine "AT Declaration" darf nicht in Kombination mit "VAR RETAIN" oder "VAR PERSISTENT verwendet werden!

Siehe im folgenden:

- Retain-Variablen
- Persistente Variablen
- Übersichtstabelle zum Verhalten

Retain-Variablen

Als "Retain-Variablen" deklarierte Variablen werden zielsystemabhängig, aber typischerweise in einem **eigenen Speicherbereich** verwaltet. Sie werden im »PLC Designer«-Projekt in ihrer Deklaration in einem Baustein bzw. in einer globalen Variablenliste mit dem Schlüsselwort "**RETAIN**" gekennzeichnet.

Beispiel:

```
VAR RETAIN
 iRem1 : INT; (* 1. Retain variable*)
END_VAR
```

Retain-Variablen behalten ihren Wert nach einem unkontrolliertem Beenden bzw. nach Online-Befehl 'Reset Warm <Applikation>', wie auch nach normalem Aus- und Einschalten (Reboot) der Steuerung. Bei erneutem Start des Programms wird mit den gespeicherten Werten weitergearbeitet. Alle anderen Variablen werden in diesem Fall neu initialisiert, entweder mit ihren initialisierten Werten oder mit den Standardinitialisierungen.

Anwendungsbeispiel: Ein Stückzähler in einer Fertigungsanlage, der nach einem Stromausfall weiter zählen soll.

Retain-Variablen werden allerdings bei 'Reset (Ursprung)' und - im Gegensatz zu persistenten Variablen - bei 'Reset (kalt)' und einem erneuten Programm-Download neu initialisiert.

Die Retain-Eigenschaft kann mit der Persistent-Eigenschaft kombiniert werden.. Sehen Sie hierzu unten die Übersichtstabelle.

Hinweis:
Wenn eine lokale Variable in einem Programm als RETAIN deklariert ist, wird genau diese Variable im Retain-Bereich gespeichert (wie eine globale Retain-Variable).

Wenn eine lokale Variable in einem Funktionsblock als RETAIN deklariert ist, wird die komplette Instanz dieses Funktionsblocks im Retain-Bereich gespeichert (alle Daten des Bausteins), wobei jedoch nur die deklarierte Retain-Variable als solche behandelt wird.

Wenn eine lokale Variable in einer Funktion als RETAIN deklariert ist, hat dies keine Auswirkung. Die Variable wird nicht im Retain-Bereich gespeichert! Wird eine lokale Variable in einer Funktion als PERSISTENT deklariert, bleibt dies ebenfalls ohne Wirkung!

Persistente Variablen

Persistente Variablen werden mit dem Schlüsselwort "PERSISTENT" (**VAR_GLOBAL PERSISTENT**) gekennzeichnet. Sie werden nur bei Reset (Ursprung) der Steuerung neu initialisiert. Im Unterschied zu Retain-Variablen behalten sie ihren Wert nach einem Download. Ein Anwendungsbeispiel für "persistente Retain-Variablen" wäre ein Betriebsstundenzähler, der nach einem Stromausfall oder Download weiter zählen soll. Siehe unten die Übersichtstabelle.

Persistente Variablen werden folgendermaßen - und damit **anders als in »PLC Designer« V2.3** - behandelt:

Persistente Variablen können NUR in einer **speziellen globalen Variablenliste** des Objekttyps 'Persistente Variablen' deklariert werden, die zu einer Applikation gehört. Es gibt nur EINE solche Liste pro Applikation.

Note: Ab V3.3.0.1 bewirkt eine Deklaration mit "VAR_GLOBAL PERSISTENT" dasselbe wie eine Deklaration mit "VAR_GLOBAL PERSISTENT RETAIN" oder "VAR_GLOBAL RETAIN PERSISTENT".

Wie Retain-Variablen werden die Persistenten Variablen in einem eigenen Speicherbereich verwaltet.

Beispiel:

```
VAR GLOBAL PERSISTENT RETAIN
 iVarPers1 : DINT; (* 1. Persistent+Retain Variable App1 *)
 bVarPers : BOOL;  (* 2. Persistent+Retain Variable App1 *)
END_VAR
```

Hinweis: Aktuell sind nur globale persistente Variablen möglich.

Vom Zielsystem muss pro Applikation ein **separater Speicherbereich** für die persistente Variablenliste bereitgestellt werden.

Bei jedem erneuten **Laden der Applikation** wird die persistente Variablenliste auf der Steuerung mit der des Projekts verglichen. Die Variablenliste auf der Steuerung wird dabei u.a. auch über den Applikationsnamen identifiziert. Bei **Inkonsistenzen** wird der Benutzer vor dem Download zu einer **Re-Initialisierung** aller persistenten Variablen aufgefordert. Inkonsistenz entsteht beispielsweise durch Umbenennen oder Löschen oder sonstige Veränderung bestehender persistenter Variablendeklarationen.

Hinweis: Also sollte jede Änderung im Deklarationsteil der Persistenten Variablenliste und die Auswirkung einer Re-Initialisierung, die daraufhin angefragt wird, sorgfältig überlegt werden!

Neue Deklarationen können nur am Ende der Liste hinzugefügt werden, werden jedoch beim Laden als neu erkannt und fordern kein Re-Initialisieren der gesamten Liste.

Übersichtstabelle zum Verhalten remanenter Variablen

x = Wert bleibt erhalten - = Wert wird neu initialisiert

nach Online-Befehl	VAR	VAR RETAIN	VAR PERSISTENT VAR RETAIN PERSISTENT VAR PERSISTENT RETAIN
Reset Warm <applikation>	-	x	x
Reset Kalt <applikation>	-	-	x
Reset Ursprung <applikation>	-	-	-
Laden (=Download) <applikation>	-	-	x
Online Change <applikation>	x	x	x
Reboot Steuerung	-	x	x

16.1.20 Konstanten - CONSTANT, Getypte Konstanten

Konstanten werden mit dem Schlüsselwort **CONSTANT** gekennzeichnet. Sie können lokal oder global deklariert werden.

Syntax:

```
VAR CONSTANT
 <Identifier>:<Type> := <initialization>;
END_VAR
```

Beispiel:

```
VAR CONSTANT
 c_iCon1:INT:=12; (* 1. Constant*)
END_VAR
```

In der Beschreibung der Operanden finden Sie eine Liste möglicher Konstanten.

Beachten Sie auch die Möglichkeit, getypte Konstanten zu verwenden:

Getypte Konstanten (Typed Literals)

Normalerweise wird beim Rechnen mit IEC-Konstanten der kleinstmögliche Datentyp verwendet. Soll ein anderer Datentyp verwendet werden, kann dies mithilfe von getypten Konstanten (Typed Literals) erreicht werden, ohne dass die Konstante explizit deklariert werden muss. Die Konstante wird hierbei mit einem Präfix versehen, welches den Typ festlegt:

Die Schreibweise ist wie folgt

Syntax:

<Type>#<Literal>

<Type> gibt den gewünschten Datentyp an, mögliche Eingaben: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. Der Typ muss in Großbuchstaben geschrieben werden.

<Literal> gibt die Konstante an. Die Eingabe muss zum unter <Type> angegebenen Datentypen passen.

Beispiel:

```
iVar1:=DINT#34;
```

Kann die Konstante nicht ohne Datenverlust in den Zieltyp überführt werden, so wird eine Fehlermeldung ausgegeben.

Getypte Konstanten können überall dort verwendet werden, wo normale Konstanten verwendet werden können.

Konstanten im Online-Modus

Wenn die Standardeinstellung Konstanten ersetzen aktiviert ist, werden Konstanten im Online-Modus im Deklarations- oder Überwachenfenster durch ein in Spalte Wert vorangestelltes Symbol gekennzeichnet. In diesem Fall kann nicht auf sie zugegriffen werden, beispielsweise durch Forcen oder Schreiben.

16.1.21 Variablenkonfiguration - VAR_CONFIG

Die "Variablenkonfiguration" kann verwendet werden, um **Funktionsblock-Variablen** auf das Prozessabbild zu "mappen", d.h. die Variablen den Geräte I/Os zuzuweisen, ohne die exakte Adresse bereits bei der Variablendeklaration im Funktionsblock angeben zu müssen. Die Zuweisung der Adresse erfolgt in diesem Fall zentral für alle Funktionsblock-Instanzen der Applikation in einer **globalen VAR_CONFIG-Liste**.

Zu diesem Zweck können den Funktionsblock-Variablen in der Deklaration zwischen den Schlüsselwörtern VAR und END_VAR **"unvollständige" Adressen** zugewiesen werden. Diese Adressen werden mit einem "*" gekennzeichnet.

Syntax:

<Bezeichner> AT %<I|Q>* : <Datentyp>;

Beispiel für die Zuweisung von unvollständig definierten Adressen:

```
FUNCTION_BLOCK locio
VAR
 xLocIn AT %I*: BOOL := TRUE;
 xLocOut AT %Q*: BOOL;
END_VAR
```

Hier werden zwei lokale I/O-Variablen definiert, eine lokale Eingabe- (%I*) und eine lokale Ausgabevariable (%Q*).

Die **endgültige Definition** der Adressen wird dann in der "Variablenkonfiguration" in einer Globalen Variablenliste vorgenommen:

Dazu fügen Sie mit Hilfe des Befehls Objekt hinzufügen ein Objekt vom Typ 'Globale Variablenliste' (GVL) im Gerätefenster ein. In dieser GVL geben Sie die Deklarationen der Instanzvariablen mit den exakten Adressen zwischen den Schlüsselwörtern **VAR_CONFIG** und **END_VAR** ein.

Die Instanzvariablen müssen mit dem vollständigen Instanzpfad angegeben werden, wobei die einzelnen POU- und Instanznamen durch jeweils einen Punkt getrennt sind. Die Deklaration muss eine Adresse enthalten, deren Klasse (Eingang/Ausgang) mit der der unvollständig angegebenen (%I*, %Q*) im Funktionsblock übereinstimmt. Der Datentyp muss ebenfalls übereinstimmen.

Syntax:

<instance variable path> AT %<I|Q><location> : <data type>;

Konfigurationsvariablen, deren Instanzpfad nicht gültig ist, weil die Instanz nicht existiert, werden als Fehler gemeldet. Umgekehrt wird auch ein Fehler ausgegeben, wenn keine Adresskonfiguration für eine Instanzvariable vorliegt, die mit einer unvollständigen Adresse deklariert ist.

Beispiel für eine Variablenkonfiguration

Folgende Verwendung des Funktionsblocks locio, siehe Beispiel oben, liegt in einem Programm vor:

```
PROGRAM PLC_PRG
VAR
locioVar1: locio;
locioVar2: locio;
END_VAR
```

Dann würde eine korrekte Variablenkonfiguration beispielsweise folgendermaßen aussehen:

```
VAR_CONFIG
PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;
PLC_PRG.locioVar1.xLocOut AT %QX0.0 : BOOL;
PLC_PRG.locioVar2.xLocIn AT %IX1.0 : BOOL;
PLC_PRG.locioVar2.xLocOut AT %QX0.3 : BOOL;
END_VAR
```

Hinweis: Änderungen von Variablen, die direkt den I/O-Adressen zugewiesen sind, werden unmittelbar im Prozessabbild angezeigt, während Änderungen von Variablen, die über eine Variablenkonfiguration "gemappt" sind, erst am Ende der zuständigen Task angezeigt werden.

16.1.22 Benutzerdefinierte Datentypen

Neben den Standard-Datentypen können auch benutzerdefinierte verwendet werden.

Zur Deklaration und Initialisierung sehen Sie bitte die Beschreibung zu DUT (Data Unit Type).

16.1.23 Erweiterbare Funktionen - PARAMS

(noch nicht implementiert)

16.1.24 FB_init, FB_reinit Methoden

FB_init

Die Methode "FB_init" ersetzt den in »PLC Designer« V2.3 verwendeten INI-Operator.

Eine Methode mit Namen "FB_init" ist eine spezielle Methode für einen Funktionsblock. Sie kann explizit deklariert werden, wird aber immer auch implizit angelegt. So kann sie für jeden Funktionsblock angesprochen werden.

Die FB_init-Methode enthält **Initialisierungs-Code** für den Funktionsblock, gemäß der Deklarationen im Deklarationsteil des Funktionsblocks. Wenn die init-Methode auch explizit deklariert wird, wird der implizite Initialisierungs-Code in die explizit angelegte Methode eingefügt. Der Programmierer kann dann weiteren Initialisierungs-Code hinzufügen.

Hinweis Wenn die Ausführung den benutzerdefinierten Initialisierungs-Code erreicht, ist der Funktionsblock bereits über den impliziten Initialisierungs-Code vollständig initialisiert.

Die FB_init-Methode wird nach einem Download für alle deklarierten Instanzen aufgerufen. Achtung: Bei Online Change werden die letzten Werte die Initialisierungswerte überschreiben!

Zur Aufrufreihenfolge im Falle von Vererbung siehe unter: FB_exit, Ausführungsreihenfolge; siehe außerdem die Möglichkeit, eine Funktionsbausteinmethode automatisch nach der Initialisierung über FB_init aufrufen zu lassen: attribute call_after_init.

Interface der FB_init-Methode:

```
METHOD fb_init : BOOL
VAR_INPUT
  bInitRetains : BOOL; // bei TRUE werden die Retain-Variablen initialisiert (Warmstart / Kaltstart)
  bInCopyCode : BOOL;  // bei TRUE wird die Instanz anschließend in den Copy-Code kopiert (Online Change)
END_VAR
```

Der Rückgabewert wird nicht verwendet.

Hinweis: Beachten Sie auch die mögliche Verwendung einer "fb_exit"-Methode und die daraus resultierende Abarbeitungsreihenfolge: Siehe dazu: FB_exit.

Benutzerdefinierte Eingabe:

In einer FB_init-Methode können zusätzliche Eingänge definiert werden. Diese müssen in der Deklaration der Funktionsblock-Instanz zugewiesen werden.

Beispiel für eine init-Methode für einen Funktionsblock "serialdevice"

```
METHOD fb_init : BOOL
VAR_INPUT
  bInitRetains : BOOL; (* Initialisierung der Retain-Variablen *)
  bInCopyCode : BOOL;  (* Instanz wird in den Copy-Code kopiert *)
  nCOMnum : INT;      (* zusätzlicher Eingang: Nummer des COM-Interfaces, auf das gehört werden soll *)
END_VAR
```

Deklaration des Funktionsblocks "serialdevice":

```
COM1 : serialdevice(nCOMnum:=1);
COM0 : serialdevice(nCOMnum:=0);
```

FB_reinit

Wenn eine Methode mit Namen "FB_reinit" für eine Funktionsblock-Instanz deklariert wird, wird diese aufgerufen, wenn die **Instanz kopiert** wird (was beispielsweise bei einem Online Change nach Änderungen an der Funktionsblock-Deklaration der Fall ist). Die Methode reinitialisiert das über den Copy-Code neu erzeugte Instanz-Modul. Eine Reinitialisierung kann erwünscht sein, weil nach dem Kopieren die Daten der ursprünglichen Instanz auf die neu angelegte Instanz geschrieben werden, aber die ursprünglichen Werte erwünscht sind. Zu beachten ist, dass dir FB_reinit Methode im Unterschied zu FB_init **explizit deklariert werden muss**. Wenn eine Reinitialisierung der Basisimplementierung des Funktionsblocks gewünscht ist, muss FB_reinit explizit für diesen Baustein aufgerufen werden.

Die FB_reinit-Methode hat keine Eingänge.

Zur Aufrufreihenfolge im Falle von Vererbung siehe unter: FB_exit, Ausführungsreihenfolge

16.1.25 FB_exit Methode

Eine spezielle Methode für einen Funktionsblock ist eine Methode mit Namen "FB_exit". Sie muss explizit deklariert werden, es gibt keine implizite Deklaration. Die exit-Methode - wenn vorhanden - wird für alle deklarierten Instanzen des Funktionsblocks vor einem erneuten **Download**, bei einem **Reset** oder während **Online Change** für alle neuen oder gelöschten Instanzen aufgerufen.

Schnittstelle der FB_exit Methode: Es gibt nur einen obligatorischen Parameter:

```
METHOD fb_exit : BOOL
VAR_INPUT
 bInCopyCode : BOOL; // bei TRUE wird die exit-Methode aufgerufen, um die Instanz zu verlassen, die hinterher kopiert wird (Online Change).
END_VAR
```

Beachten Sie auch die "FB_init" Methode und die folgende **Ausführungsreihenfolge**:

1. Exit Methode: alte Instanz verlassen: old_inst.fb_exit(bInCopyCode := TRUE);
2. Init Methode: neue Instanz initialisieren: new_inst.fb_init(bInitRetains := FALSE, bInCopyCode := TRUE);
3. Kopieren der Funktionsblock-Werte (Copy-Code): copy_fub(&old_inst, &new_inst);

Im Falle von **Vererbung** gilt dabei außerdem folgende Aufrufreihenfolge (dabei ist für die in dieser Auflistung als Beispiel genannten POUs folgendes angenommen: "SubFB EXTENDS MainFB" und "SubSubFB EXTENDS SubFB"):

fbSubSubFb.FB_Exit(...);

fbSubFb.FB_Exit(...);

fbMainFb.FB_Exit(...);

fbMainFb.FB_Init(...);

fbSubFb.FB_Init(...);

fbSubSubFb.FB_Init(...);

für FB_reinit:

fbMainFb.FB_reinit(...);

fbSubFb.FB_reinit(...);

fbSubSubFb.FB_Init(...);

16.1.26 Pragma Anweisungen

Eine Pragma-Anweisung wird verwendet, um die Eigenschaften einer oder mehrerer Variablen bezüglich der **Kompilierung** oder des Prä-Kompilierungsprozesses (**Präprozessor**) zu beeinflussen. Das heißt, ein Pragma beeinflußt die **Code-Generierung**.

Beispielsweise kann es bestimmen, ob eine Variable initialisiert, im Online-Modus angezeigt, zur Symbolliste hinzugefügt oder im Bibliotheksverwalter unsichtbar gehalten wird. Meldungsausgaben während des Übersetzungsvorgangs können erzwungen werden und bedingte Pragmas können verwendet werden, die festlegen, wie eine Variable unter bestimmten Bedingungen interpretiert werden sollen. Diese bedingten Pragmas können auch als "Compiler-Definitionen" in den Übersetzungs-Eigenschaften eines Objekts eingetragen werden.

Ein Pragma kann in einer separaten Zeile oder zusammen mit Code-Text in einer Implementations- oder Deklarationszeile eingefügt werden. Innerhalb des FUP/KOP/AWL-Editors muss zunächst das Kommando 'Sprungmarke einfügen' aufgerufen werden und anschließend der Standardeintrag "Label:" im Textfeld der Marke durch die entsprechende Pragma Anweisung ersetzt werden. Sofern Sie sowohl eine Marke als auch ein Pragma verwenden wollen, tragen Sie zunächst das Pragma ein und dann die Sprungmarke.

Eine Pragma-Anweisung wird in geschweiften Klammern angegeben, Groß-/Kleinschreibung spielt keine Rolle:

Syntax:

{ <instruction text> }

Die öffnende Klammer darf unmittelbar auf einen Variablennamen folgen. Öffnende und schließende Klammer müssen sich in der selben Zeile befinden.

Abhängig von Typ und Inhalt eines Pragmas wirkt dieses entweder auf die Zeile, in der es steht, oder auf alle nachfolgenden Zeilen bis es mit einem entsprechenden Pragma wieder aufgehoben wird, oder bis dasselbe Pragma mit anderen Parametern ausgeführt, oder das Ende der Datei erreicht wird. Als Datei wird hierbei verstanden: Deklarationsteil, Implementationsteil, Globale Variablenliste, Typdeklaration.

Kann der Compiler den Anweisungstext nicht sinnvoll interpretieren, so wird das gesamte Pragma wie ein Kommentar behandelt und überlesen. Es wird jedoch eine Warnung ausgegeben.

Mögliche Pragma-Typen:

- Meldungs-Pragmas
- Obsolete-Pragma
- Attribut-Pragma
- Bedingte Pragmas
- Pragmas für Symbolkonfiguration

noch nicht implementiert: Pragmas wie in V2.3

Meldungs-Pragmas

Meldungs-Pragmas können verwendet werden, um die Ausgabe von Meldungen im Meldungsfenster während des Übersetzungsvorgangs ("Build" des Projekts) zu erzwingen.

Die Pragma-Anweisung kann in einer separaten oder in einer bestehenden Zeile im Texteditor einer POU. Meldungs-Pragmas, die in aktuell nicht "defined" Sektionen des Implementierungs-Codes stehen, werden beim Kompilieren des Projekts nicht berücksichtigt. Siehe hierzu das Beispiel innerhalb der Beschreibung des Bedingten Pragmas "defined (*identifier*)".

Es gibt vier Typen von Meldungs-Pragmas:

Pragma	Meldungstyp
{text '*textstring*'}	**Text**: Der angegebene Text *textstring* wird ausgegeben.
{info '*textstring*'}	**Information**: Der angegebene Text *textstring* wird ausgegeben.
{warning digit '*textstring*'}	**Warnung**: Der angegebene Text *textstring* wird ausgegeben. Noch nicht implementiert: Die angegebene Zahl zeigt die Warnstufe (zwischen 1 und 5) an. Im Unterschied zu einem "datentyp-globalen" Obsolete-Pragma wird die Warnung lokal für die aktuelle Position definiert.
{error '*textstring*'}	**Fehler**: Der angegebene Text *textstring* wird ausgegeben.

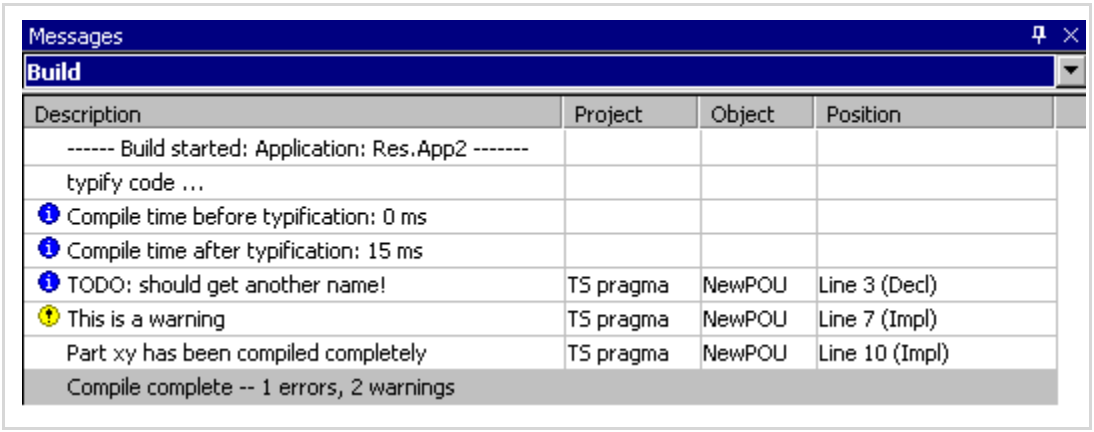
Hinweis: Im Fall von Meldungstyp Information, Warnung und Fehler gelangt man über den Befehl 'Nächste Meldung' an die Quellposition der Meldung - d.h. die Position, an der das Pragma in der POU platziert ist.

Beispiel

Deklaration und Implementation im ST-Editor:

```
VAR
ivar : INT; {info 'TODO: should get another name'}
bvar : BOOL;
arrTest : ARRAY [0..10] OF INT;
i:INT;
END_VAR
arrTest[i] := arrTest[i]+1;
ivar:=ivar+1;
{warning 'This is a warning'}
{text 'Part xy has been compiled completely'}
```

Ausgabe im Meldungsfenster:



Attribut-Pragmas

Attribut-Pragmas

Attribut-Pragmas können einer Signatur hinzugefügt werden um die Kompilierung bzw. Vorübersetzung zu beeinflussen.

So gibt es benutzerdefinierte Attribute, die in Verbindung mit bedingten Pragmas verwendet werden können, aber auch die unten aufgelisteten vordefinierten Attribut-Pragmas:

- Attribut 'displaymode'
- Attribut 'ExpandFully'
- Attribut 'global_init_slot'
- Attribut 'hide'
- Attribut 'hide_all_locals'
- Attribut 'initialize_on_call'
- Attribut 'Init_Namespace'
- Attribut 'init_on_onlchange'
- Attribut 'instance-path'
- Attribut 'linkalways'
- Attribut 'Monitoring'
- Attribut 'no_check'
- Attribut 'no_copy'
- Attribut 'No-Exit'
- Attribut 'noinit'
- Attribut 'no_virtual_actions'
- Attribut 'obsolete'

- Attribut 'pack_mode'
- Attribut 'parameterstringof'
- Attribut 'qualified_only'
- Attribut 'reflection'
- Attribut 'symbol'

Benutzerdefinierte Attribute

Ein beliebiges benutzer- oder applikationsdefiniertes Attribut kann einem Baustein, einer Typdefinition oder einer Variablen zugewiesen werden. Dieses Attribut kann vor der Übersetzung des Projekts mit Hilfe von bedingten Pragmas abgefragt werden.

Syntax:

{attribute *'attribute'*}

Dieses Pragma wirkt immer auf die aktuelle Zeile oder, wenn es in einer separaten Zeile steht, auf die nachfolgende.

Ein Attribut kann folgenden Objekten zugewiesen werden:

- POUs, Aktionen Beispiel

Attribut "vision" für Funktion fun1:

```
{attribute 'vision'}
FUNCTION fun1 : INT
VAR_INPUT
i : INT;
END_VAR
VAR
END_VAR
```

- Variablen Beispiel

Attribut "DoCount" für Variable ivar :

```
PROGRAM PLC_PRG
VAR
ivar:INT; {attribute 'DoCount'};
bvar:BOOL;
END_VAR
```

- Typen Beispiel

Attribut "aType" für Datentyp DUT_1:

```
{attribute 'aType'}
TYPE DUT_1 :
STRUCT
a:INT;
b:BOOL;
END_STRUCT
END_TYPE
```

Attribut call_after_init

Mit Hilfe des Pragmas {attribute call_after_init} kann eine Methode definiert werden, die implizit nach der Initialisierung einer Funktionsbausteininstanz aufgerufen werden soll. Dazu muss das Attribut sowohl dem Funktionsbaustein als auch der Methode hinzugefügt werden (aus Performanzgründen). Die Methode muss nach FB_Init, und nachdem die Variablenwerte eines Initialisierungsausdrucks in der Instanzdeklaration gültig wurden, aufgerufen

werden.

Diese Funktionalität wird ab Compiler Version >= 3.4.1.0 unterstützt.

Syntax:

{attribute 'attribute call_after_init'}

Beispiel:

Mit der folgenden Definition:

```
{attribute 'call_after_init'}
FUNCTION_BLOCK FB
... <functionblock definition>

{attribute 'call_after_init'}
METHOD FB_AfterInit
... <method definition>
```

... wird eine Deklaration wie diese:

```
inst : FB := (in1 := 99);
```

... in folgende Code-Abarbeitung umgesetzt:

```
inst.FB_Init();
inst.in1 := 99;
inst.FB_AfterInit();
```

Somit kann also in FB_AfterInit auf die benutzerdefinierte Initialisierung reagiert werden.

Attribut displaymode

Mit Hilfe des Pragmas {attribute 'displaymode'} kann der Darstellungsmodus einer einzelnen Variable definiert werden. Diese Festlegung überschreibt die globale Einstellung für die Darstellung aller Monitoring-Variablen, die über die Befehle im Untermenü Darstellung (standardmäßig im Online-Menü) erfolgt.

Das Pragma muss in der Zeile oberhalb der Zeile stehen, die die Variablendeklaration enthält.

Syntax:

{attribute 'displaymode’:=<displaymode>}

Die folgenden Definitionen sind möglich:

- für Darstellung in Binärformat:

```
{attribute 'displaymode':='bin'}
{attribute 'displaymode':='binary'}
```

- für Darstellung in Dezimalformat:

```
{attribute 'displaymode':='dec'}
{attribute 'displaymode':='decimal'}
```

- für Darstellung in Heximalformat:

```
{attribute 'displaymode':='hex'}
{attribute 'displaymode':='hexadecimal'}
```

Beispiel:

```
VAR
  {attribute 'displaymode':='hex'}
  dwVar1: DWORD;
END_VAR
```

Attribut ExpandFully

Mit Hilfe des Pragmas {attribute 'ExpandFully'} können die Komponenten eines Arrays, das als Eingabevariable für referenzierte Visualisierungen verwendet wird, im Eigenschaften-Dialog der Visualisierung sichtbar gemacht werden.

Syntax:

{attribute 'ExpandFully'}

Example:

Visualisierung *visu* soll in einen Frame innerhalb von Visualisierung *visu_main* eingefügt werden.

arr ist als Eingabevariable im Schnittstellen Editor von *visu* definiert und wird somit später für Zuweisungen im Eigenschaften-Dialog des Frames in *visu_main* bereitstehen.

Um auch die einzelnen Komponenten des Arrays in diesem Eigenschaften-Dialog verfügbar zu bekommen, muss das Attribut 'ExpandFully' im Schnittstellen Editor von *visu* direkt vor *arr* eingefügt werden.

Deklaration im Schnittstellen Editor von *visu*:

```
VAR_INPUT
{attribute 'ExpandFully'}
  arr : ARRAY[0..5] OF INT;
END_VAR
```

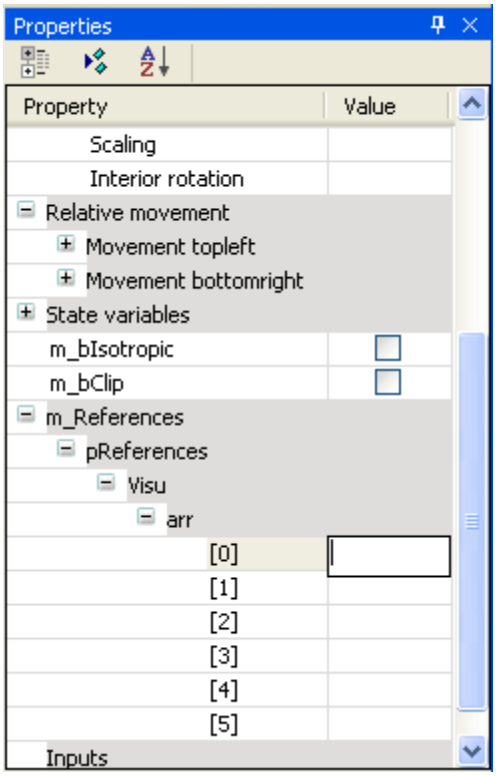


Abb. 536 Eigenschaften-Dialog des Frames in visu_main

Attribut Global_init_slot

Das Pragma **{attribute 'global_init_slot'}** kann nur auf Signaturen angewandt werden. Die Reihenfolge bei der Initialisierung von Variablen aus globalen Variablenlisten ist nicht von vornherein festgelegt. Manchmal jedoch ist das Vorschreiben einer solchen Reihenfolge notwendig, beispielsweise wenn die Variablen einer Liste von Variablen einer anderen Liste abhängen. In diesem Fall kann das Pragma zur Festlegung der Reihenfolge bei der globalen Initialisierung genutzt werden.

Syntax:

{attribute 'global_init_slot' := '<Value>'}

Der Platzhalter <Value> muss durch einen ganzzahligen Wert ersetzt werden, welcher den Stellenwert in der Reihenfolge der Initialisierung festsetzt. Der Standardwert ist 50000. Ein niedrigerer Wert bewirkt eine frühere Initialisierung. Tragen mehrere Signaturen denselben Stellenwert für das Attribut 'global_init_slot', so bleibt die Reihenfolge ihrer Initialisierung unbestimmt.

Beispiel:

Das Beispielprojekt enthält zwei globale Variablenlisten GVL_1 und GVL_2:

Die globale Variable "A" ist Teil der globalen Variablenliste GVL_1:

```
{attribute 'global_init_slot' := '300'}
VAR_GLOBAL
A : INT:=1000;
END_VAR
```

Der Initialisierungswert der Variablen "B" und "C" von GVL_2 sind abhängig von der Variablen "A".

```
{attribute 'global_init_slot' := '350'}
VAR_GLOBAL
B : INT:=A+1;
C : INT:=A-1;
END_VAR
```

Wird das Attribut 'global_init_slot' der globalen Variablenliste GVL_1 auf 300 gesetzt, also auf den im Beispiel niedrigsten Initialisierungswert, so ist sicher gestellt, dass der Ausdruck "A+1" wohldefiniert ist zum Zeitpunkt der Initialisierung von "B".

Attribut Hide

Mit Hilfe des Pragmas {attribute 'hide'} kann verhindert werden, dass Variablen oder sogar ganze Signaturen sichtbar sind innerhalb der "Komponenten Auflisten" Funktionalität, innerhalb des Eingabe-Assistenten oder des Deklarationsteils im Online-Modus. Dabei wird nur die direkt auf das Pragma folgende Variable unsichtbar gemacht.

Syntax:

{attribute 'hide'}

Um alle lokalen Variablen einer Signatur zu verstecken, benutzen Sie DIESES Attribut.

Example:

Der Funktionsblock myPOU sei mit Verwendung des Attributs implementiert:

```
FUNCTION_BLOCK myPOU
VAR_INPUT
a:INT;
{attribute 'hide'}
a_invisible: BOOL;
a_visible: BOOL;
END_VAR
VAR_OUTPUT
b:INT;
END_VAR
VAR
END_VAR
```

Im Hauptprogramm werden zwei Instanzen des Funktionsblocks myPOU definiert:

```
PROGRAM PLC_PRG
VAR
POU1, POU2: myPOU;
END_VAR
```

Während nun beispielsweise ein Eingabewert für POU1 implementiert wird, zeigt die "Komponenten Auflisten" Funktion, die sich beim Tippen von "POU1." (im Implementationsteil von PLC_PRG) öffnet, die Variablen "a", "a_visible" und "b" an, nicht aber die versteckte Variable "a_invisible".

Attribut Hide_all_locals

Mit Hilfe des Pragmas {attribute 'hide_all_locals'} können lokale Variablen einer Signatur versteckt werden innerhalb der Anzeige der "Komponenten Auflisten" Funktionalität, des Eingabe-Assistenten des Online-Monitoring im Deklarationsteil. Die Wirkung des Attributs ist identisch mit der Anwendung des Attributs 'hide' auf jede lokale Variable einzeln.

Syntax:

{attribute 'hide_all_locals'}

Beispiel:

Der Funktionsblock myPOU ist unter Zuhilfenahme des Attributs implementiert:

```
{attribute 'hide_all_locals'}
FUNCTION_BLOCK myPOU
VAR_INPUT
a:INT;
END_VAR
VAR_OUTPUT
b:BOOL;
END_VAR
VAR
c,d:INT;
END_VAR
```

Im Hauptprogramm seien zwei Instanzen des Funktionsblocks myPOU definiert:

```
PROGRAM PLC_PRG
VAR
POU1, POU2: myPOU;
END_VAR
```

Während nun beispielsweise ein Eingabewert für POU1 implementiert wird, zeigt "Komponenten Auflisten" Funktion, die sich beim Tippen von "POU1." (im Implementationsteil von PLC_PRG) öffnet, die Variablen "a" und "b" an, nicht aber die versteckten lokalen Variablen "c" oder "d".

Attribut Initialize_on_call

Das Pragma **{attribute 'initialize_on_call'}** kann auf Eingangsvariablen angewandt werden. Ein Eingabegröße eines Funktionsblocks, die mit diesem Attribut versehen ist, wird bei jedem Aufruf des Funktionsblocks initialisiert. Erwartet ein Eingang einen Zeiger und wurde dieser im Zuge eines Online-Change entfernt, so wird dieser Eingang auf NULL gesetzt.

Syntax:

{attribute 'initialize_on_call'}

Attribute Init_Namespace

Eine Variable vom Typ STRING oder WSTRING, die mit dem Pragma **{attribute 'init_namespace'}** versehen ist, wird mit dem aktuellen Namensraum initialisiert. Die Verwendung des Pragmas setzt die zusätzliche Verwendung von Attribut 'noinit' für die String-Variable voraus.

Syntax:

{attribute 'init_namespace'}

Beispiel:

Der Funktionsbaustein POU ist mit den nötigen Attributen versehen:

```
{attribute 'reflection'}
FUNCTION_BLOCK POU
VAR_OUTPUT
{attribute 'no_init'}
{attribute 'instance-path'}
myStr: STRING;
END_VAR
```

Innerhalb des Hauptprogramms PLC_PRG ist eine Instanz "fb" des Funktionsbausteins POU definiert:

```
PROGRAM PLC_PRG
VAR
fb:POU;
newString: STRING;
END_VAR
newString:=fb.myStr;
```

Die Variable myStr wird mit dem aktuellen Namensraum initialisiert, z.B. "PLCWinNT.Application.PLC_PRG.fb". Dieser Wert wird newString im Hauptprogramm zugewiesen.

Attribut Init_On_Onlchange

Wird das Pragma **{attribute 'init_on_onlchange'}** auf eine Variable angewendet, so wird diese im Zuge von jedem Online Change initialisiert.

Syntax:

{attribute 'init_on_onlchange' }

Attribut Instance-path

Das Pragma **{attribute 'instance-path'}** kann auf eine lokale String-Variablen angewandt werden, welche in Folge mit dem Gerätebaum-Pfad der POU, zu welcher sie gehört initialisiert wird. Dies kann nützlich sein für Fehlermeldungen. Die Anwendung des Pragmas setzt die Anwendung des Attributs 'reflection' auf die zugehörige POU voraus, sowie die Anwendung des zusätzlichen Attributs 'noinit' auf die String-Variable selbst.

Syntax:

{attribute 'instance-path'}

Beispiel:

Der folgende Funktionsblock POU sei mit allen notwendigen Attributen ausgestattet:

```
{attribute 'reflection'}
FUNCTION_BLOCK POU
VAR
{attribute 'instance-path'}
{attribute 'noinit'}
str: STRING;
END_VAR
```

Im Hauptprogramm PLC_PRG sei eine Instanz "myPOU" des Funktionsblocks POU definiert:

```
PROGRAM PLC_PRG
VAR
myPOU:POU;
myString: STRING;
END_VAR
myPOU();
myString:=myPOU.str;
```

Die Initialisierung der Instanz myPOU enthält die String-Variable "str" den Pfad der Instanz myPOU, im Beispiel " PLCWinNT.Application.PLC_PRG.myPOU". Dieser Pfadwird im Hauptprogramm der Variablen myString zugewiesen.

Hinweis: Die Länge eines Strings kann beliebig definiert werden (auch >255), allerdings muss bedacht werden, dass der String (von hinten her) abgeschnitten wird, wenn er einer Variable zugewiesen ist, deren Datentyp zu klein dafür ist.

Attribut linkalways

Mit Hilfe des Pragmas {attribute 'linkalways'} wird das zugehörige Objekt beim Compiler markiert und ist damit immer in der Compiler-Information enthalten, was bedeutet, dass es immer kompiliert und auf die SPS geladen wird. Das Pragma wirkt nur für POUs und GVLs, die unterhalb einer Applikation oder in Bibliotheken unterhalb einer Applikation liegen. Die Compiler-Option **Immer binden** macht das selbe.

Syntax:

{attribute 'linkalways'}

Beim Verwenden des Symbolkonfigurationseditor wird die markiere POU als Basis für die auswählbaren Variablen der Symbolkonfiguration verwendet

Beispiel:

Die Implementierung der globale Variablenliste GVLMoreSymbols benutzt das Attribut 'linkalways':

```
{attribute 'linkalways'}
VAR_GLOBAS
g_iVar1: INT;
g_iVar2: INT;
END_VAR
```

Durch diesen Code werden die Variablen von GVLMoreSymbols als auswählbare Symbole zur Verfügung gestellt.

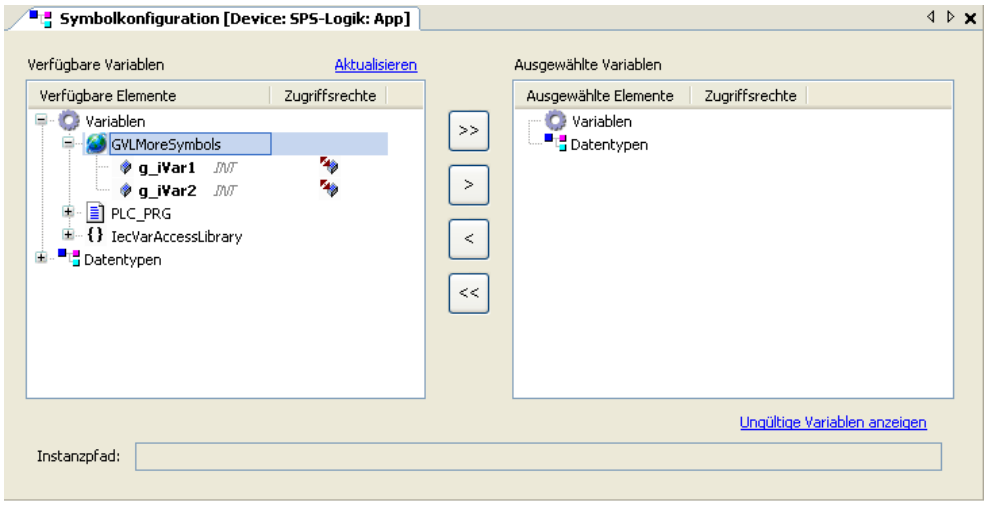


Abb. 537 Editor der Symbolkonfiguration

Attribut Monitoring

Eine Eigenschaft kann im Online-Modus entweder mit Hilfe des Inline Monitoring (noch nicht implementiert) oder mit Hilfe einer Watch-Liste überwacht werden.

Das Monitoring kann aktiviert werden, in dem das "monitoring" Attribut-Pragma in der Zeile oberhalb der Definition der Eigenschaft eingefügt wird. Dann wird in der Online-Ansicht des Bausteins, der die Eigenschaft verwendet, bzw. in einer Überwachungsliste der Name, Typ und Wert der Variablen der Eigenschaft angezeigt. Dort können auf Werte für das Forcen der Eigenschaften-Variablen vorbereitet werden.

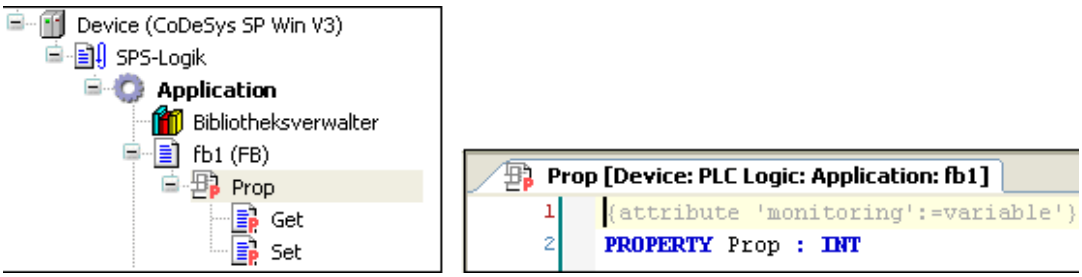


Abb. 538 Beispiel einer zum Monitoring vorbereiteten Eigenschaft

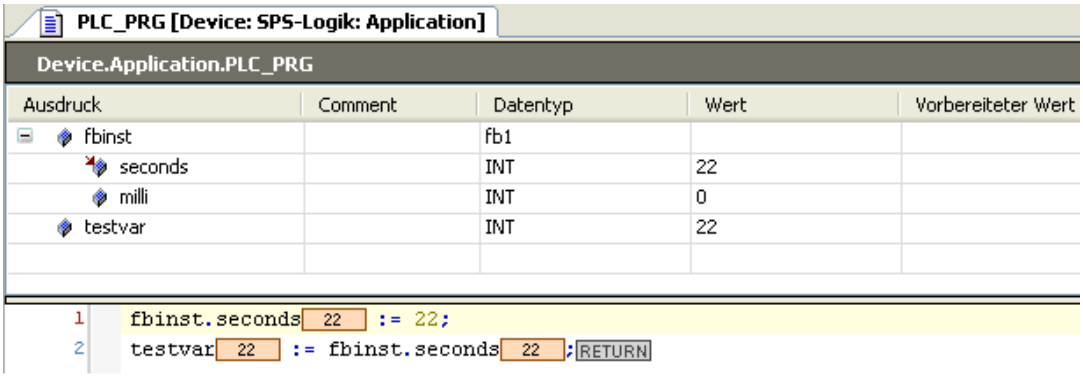


Abb. 539 Beispiel einer Monitoring-Ansicht

Es gibt zwei verschiedene Wege, den aktuellen Wert der Eigenschaften-Variablen angezeigt zu bekommen. Im jeweiligen Anwendungsfall muss sorgfältig erwogen werden, welches Attribut-Pragma geeignet ist, den gewünschten Wert anzuzeigen. Das hängt davon ab, ob innerhalb der Eigenschaft weitere Operationen mit den Variablen implementiert sind.

1. Pragma **'{attribute 'monitoring':='variable'}'**

Eine implizite Variable wird für die Eigenschaft angelegt, die immer wenn die Applikation die Set oder Get Methode aufruft, den aktuellen Eigenschaften-Wert erhält. Der Wert dieser impliziten Variable wird im Monitoring dargestellt.

Syntax:

{attribute 'monitoring':='variable'}

2. Pragma **'{attribute 'monitoring':='call'}'**

Dieses Attribut kann nur für Eigenschaften verwendet werden, die einfache Datentype oder Pointer zurückgeben, nicht für strukturierte Typen:

Der zu monitorende Wert wird durch direktes Aufrufen der Eigenschaft geholt, d.h. der Monitoring-Dienst des Laufzeitsystems ruft die Get Methode auf. Achtung, wenn in der Eigenschaft Operationen auf die Variablen implementiert sind, kann sich der Wert noch ändern!

Syntax:

{attribute 'monitoring':='call'}

Attribut No_check

Das Pragma **{attribute 'no_check'}** wird einer POU hinzugefügt um jeden Aufruf einer Check-Funktion (siehe "POUs für implizite Prüfungen") zu unterbinden.. Da die Check-Funktionen die Abarbeitungsgeschwindigkeit des Programms beeinflussen können, kann es sinnvoll sein das Attribut auf Bausteine anzuwenden, die schon geprüft sind oder oft aufgerufen werden.

Syntax:

{attribute 'no_check’}

Attribut No_Copy

Im allgemeinen erfordert ein Online Change eine Reallozierung von Instanzen bspw. einer POU. Dabei wird der Wert der in der Instanz enthaltenen Variablen kopiert.

Wird jedoch das Pragma **{attribute 'no_copy'}** auf eine Variable angewendet, so wird keine Kopie des Variablenwertes durchgeführt; die Variable wird stattdessen neu initialisiert im Zuge eines Online Changes. Das kann sinnvoll sein für eine lokale Zeigervariable, die auf eine Variable zeigt, die nun gerade durch den Online Change verschoben wurde (und somit eine veränderte Adresse besitzt).

Syntax:

{attribute 'no_copy’}

Attribut No-exit

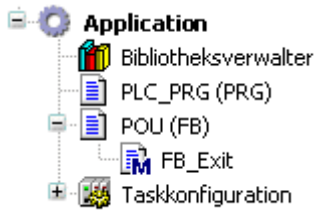
Stellt ein Funktionsblock eine Exit Methode bereit, so kann der Aufruf dieser Methode für eine spezielle Instanz des Funktionsblocks unterdrückt werden, indem das Pragma **{attribute 'no-exit'}** auf die Instanz angewendet wird.

Syntax:

{attribute 'symbol' := 'no-exit'}

Beispiel:

Dem Funktionsblock namens "POU" sei die Exit Methode "FB_Exit" hinzugefügt:



Im Hauptprogramm PLC_PRG werden zwei Instanzen des Funktionsblocks "POU" angelegt:

```
PROGRAM PLC_PRG
VAR
POU1 : POU;
{attribute 'symbol' := 'no-exit'}
POU2 : POU;
END_VAR
```

Wird der Variablen bInCopyCode innerhalb von POU1 der Wert TRUE zugewiesen, so wird die Exit Methode FB_Exit aufgerufen. Im Gegensatz dazu hat der Wert der Variablen bInCopyCode innerhalb von POU2 keinerlei Auswirkungen.

Attribut Noinit

Variablen, welche mit dem Pragma **{attribute 'no_init'}** versehen sind, werden nicht implizit initialisiert. Das Pragma bezieht sich nur auf die in direkter Folge deklarierten Variable.

Syntax:

{attribute 'no_init'}

also possible:

{attribute 'no-init'}

{attribute 'noinit'}

Beispiel:

```
PROGRAM PLC_PRG
VAR
A : INT;
{attribute 'no_init'}
B : INT;
END_VAR
```

Bei einem Reset der zugehörigen Applikation wird die Integer-Variable "A" erneut implizit mit 0 initialisiert, wohingegen die Variable "B" ihren aktuellen Wert beibehält.

Attribut No_virtual_actions

Dieses Attribut betrifft Funktionsbausteine welche von einem in SFC implementierten Funktionsblock abgeleitet werden und den grundsätzlichen SFC-Ablauf dieser Basisklasse nutzen. Die daraus aufgerufenen Aktionen zeigen dasselbe virtuelle Verhalten wie Methoden, das heißt, die Implementierungen der Aktionen in der Basisklasse können von der abgeleiteten Klasse durch eigene, spezifische Implementierungen ersetzt werden.

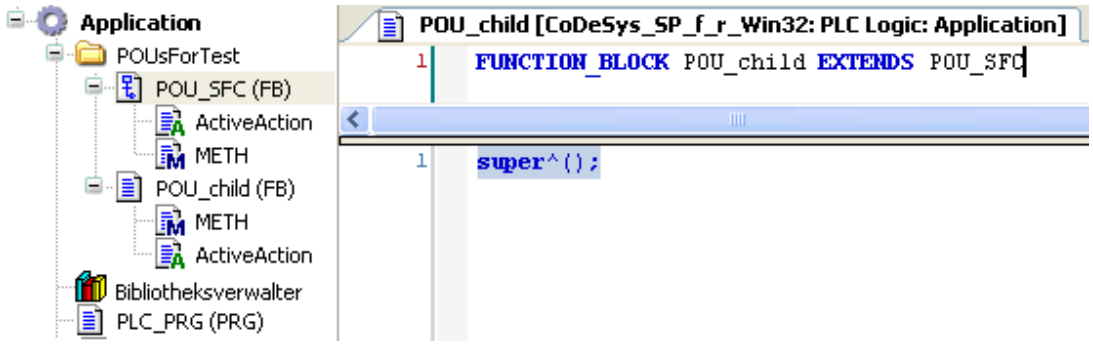
Wird der Basis Klasse jedoch das Pragma {attribute 'no_virtual_actions'} verliehen, so sind ihre Aktionen vor einem Überladen geschützt.

Syntax:

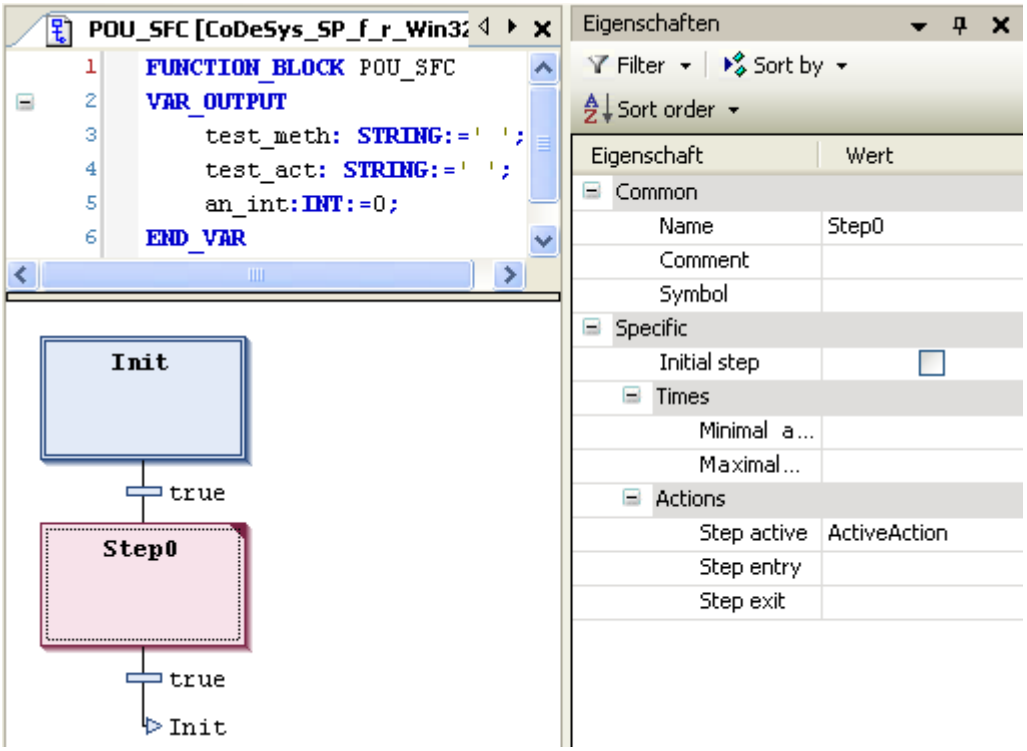
{attribute 'no_virtual_actions'}

Beispiel:

Im folgenden Beispiel bildet der Funktionsblock POU_SFC die Basisklasse für den abgeleiteten Funktionsblock POU_child.



Durch Benutzung des Schlüsselworts SUPER ruft die abgeleitete Klasse POU_child den in SFC geschriebenen Ablauf der Basisklasse auf.



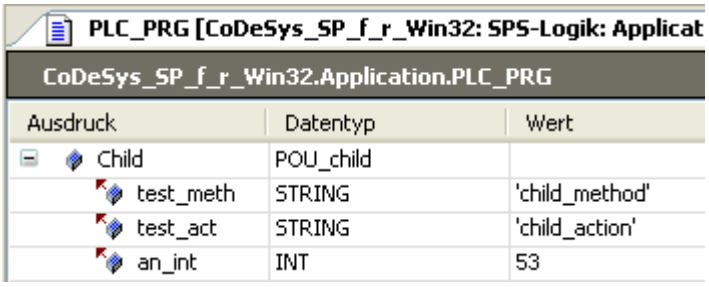
Die beispielhafte Implementierung dieses Ablauf ist beschränkt auf den Initialschritt gefolgt von einem einzigen Schritt mit angebundener Schrittaktion "ActiveAction", welche die Belegung der Ausgangsvariablen übernimmt:

```
an_int:=an_int+1;        // Zählen der Aktionsaufrufe
test_act:='father_action'; // Belegung der Stringvariablen test_act
METH();                  // Aufruf der Methode METH zur Belegung der Stringvariablen
test_meth
```

Im Fall der abgeleiteten Klasse POU_child wird die Schrittaktion ersetzt durch eine spezielle Implementierung von ActiveAction, welche sich vom Original nur durch Zuweisung des Strings 'child_action' anstelle von 'father_action' an die Variable test_act unterscheidet.

Ebenso wird die Methode METH, welche in der Basisklasse der Variablen test_meth den String "father_method" zuweist, dahingehend überschreiben, dass test_meth nun den Wert "child_method" erhält.

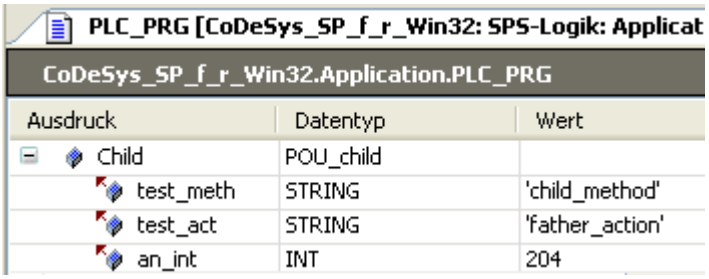
Das Hauptprogramm PLC_PRG ruft eine Instanz des Funktionsblock POU_child namens "Child" auf. Wie erwartet spiegelt der Wert der Strings den Aufruf von Aktion und Methode der abgeleiteten Klasse wider:



Wird der Basis jedoch das Attribut 'no_virtual_actions' vorangestellt

```
{attribute 'no_virtual_actions'}
FUNCTION_BLOCK POU_SFC...
```

so kann ein anderes Verhalten beobachtet werden: wohingegen für Methode METH weiterhin die Implementierung der abgeleiteten Klasse herangezogen wird, resultiert der Aufruf der Schrittaktion nun in einem Aufruf der Aktion ActiveAction der Basisklasse. Daher erhält test_act nun den Wert 'father_action':



Attribut Obsolete

Ein"Obsolete-Pragma" kann einer Datentyp-Definition hinzugefügt werden, um bei Übersetzen eine definierte Warnung auszugeben, wenn der Datentyp (Struktur, Funktionsbaustein etc.) im Projekt verwendet wird. Damit kann beispielsweise darauf hingewiesen werden , dass ein Datentyp nicht mehr gültig ist, weil sich beispielsweise eine Schnittstelle geändert hat und dies im Projekt nachgezogen werden sollte.

Im Unterschied zu einem lokal verwendeten Meldungspragma wird diese Warnung "zentral" für alle Instanzen eines Datentyps definiert.

Dieses Pragma wirkt immer auf die aktuelle Zeile oder, wenn es in einer separaten Zeile steht, auf die nachfolgende.

Syntax:

{attribute "obsolete' := *'user defined text'*}

Beispiel:

Das Attribut wird in der Definition von Funktionsblock fb1 eingefügt:

```
{attribute 'obsolete' := 'datatype fb1 not valid!'}
FUNCTION_BLOCK fb1
VAR_INPUT
i:INT;
END_VAR
...
```

Wenn fb1 als Datentyp verwendet wird, z.B. in "fbinst: fb1;" wird beim Übersetzen des Projekts die Warnung ausgegeben: "datatype fb1 not valid"'.

Attribut Pack_mode

Das Pragma **{attribute 'pack_mode'}** legt fest, wie eine Datenstruktur während der Allozierung gepackt wird. Das Attribut muss oberhalb der Datenstruktur eingefügt werden und wirkt sich auf das Packen der gesamten Struktur aus.

Syntax:

{attribute 'pack_mode' := '<Value>'}

Der in einfache Hochkommata eingeschlossene Platzhalter <Value> muss durch einer der folgenden Werte ersetzt werden:

Wert	Assoziierte Packungsart
0	aligned, d.h. es treten keine Speicherlücken auf
1	1-byte-aligned (identisch zu aligned)
2	2-byte-aligned, d.h. die maximale Größe einer Speicherlücke ist 1 byte
4	4-byte-aligned, d.h. die maximale Größe einer Speicherlücke ist 3 byte
8	8-byte-aligned, d.h. die maximale Größe einer Speicherlücke ist 7 byte

Beispiel:

```
{attribute 'pack_mode' := '1'}
TYPE myStruct:
STRUCT
 Enable: BOOL;
 Counter: INT;
 MaxSize: BOOL;
 MaxSizeReached: BOOL;
END_STRUCT
END_TYPE
```

Der Speicherbereich für eine Variable vom Datentyp myStruct wird "aligned" alloziert: Ist die Speicheradresse ihrer Komponente Enable bspw. 0x0100, dann folgt die Komponente Counter an der Adresse 0x0101, MaxSize an Adresse 0x0103 und MaxSizeReached an Adresse 0x0104. Mit 'pack_mode'=2 läge Counter bei 0x0102, MaxSize bei 0x0104 und MaxSizeReached bei 0x0105.

Hinweis: Das Attribut kann auch auf POUs angewandt werden. Aufgrund von eventuell vorhandenen internen Zeigern muss dies jedoch sorgsam gehandhabt werden.

Attribut Parameterstringof

Mit Hilfe des Attribut Pragmas {attribute 'parameterstringof'} wird der Instanzname einer Variablen einem Visualisierungsbaustein zur Verfügung gestellt.

Syntax:

{attribute 'parameterstringof' := ' <variable>'}

Beispiel:

Im Hauptprogramm sei die Instanz myDUT der benutzerdefinierten Struktur DUT angelegt:

```
PROGRAM PLC_PRG
VAR
myDUT: DUT;
END_VAR
```

Diese Instanz sei Eingabe eines Visualisierungsbausteins "Vis" (im Eingabe-Parameter "instance") welcher seinerseits von einem Frame einer anderen Visualisierung "MainVisu" referenziert wird:

Ausschnitt aus Eigenschaften eines Visualisierungselements von MainVisu

Elementname	GenElemInst_2
Clipping	☐
Show frame	☐
Scaletype	ANISOTROPIC
⊟ References	_3S.CoDeSys.VisualElem.StructuredTypeNode
⊟ References	
⊟ PLCWinNT.Application.Vis	
instance	PLC_PRG.myDut
⊟ Position	
Y	104

In dem zu "Vis" gehörenden Schnittstellen-Editor finden wir natürlich die Input/Output Variable "instance", aber auch eine weitere Input Variable genannt "instanceStr":

Vis [PLCWinNT: SPS-Logik: Application]

```
1  VAR_INPUT
2      {attribute 'parameterstringof' := 'instance'}
3      instanceStr : STRING;
4  END_VAR
5  VAR_IN_OUT
6      instance : DUT;
7  END_VAR
```

Obwohl "instanceStr" eine Input Variable ist, ist sie nicht als Eingang bei der Referenz mit aufgeführt (vergleiche oberes Bild!). Das liegt daran, dass die Variable "instanceStr" das Attribut 'parameterstringof' trägt und somit automatisch initialisiert wird mit dem Namen der Variablen, die beim Attribut angegeben wurde. In unserem Beispiel ist "instance" die zugehörige Variable, die String Variable instanceStr wird also auf "PLC_PRG.myDUT" gesetzt und kann nun innerhalb der Visualisierung "Vis" verwendet werden, beispielsweise als Textvariable für einen Platzhalter %s.

Attribut Qualified_Only

Wird das Pragma **{attribute 'qualified_only'}** einer globalen Variablenliste vorangestellt, so können die in ihr enthaltenen Variablen nur durch Angabe des globalen Variablennamens angesprochen werden, bspw. gvl.g_var. Das trifft auch auf Variablen vom Typ Enumeration zu und kann hilfreich sein um eine Verwechslung mit lokalen Variablen zu vermeiden.

Syntax:

{attribute 'qualified_only'}

Beispiel:

Die folgende globale Variablenliste "GVL" sei mit dem Attribut 'qualified_only' ausgestattet:

```
{attribute 'qualified_only'}
VAR_GLOBAL
iVar:INT;
END_VAR
```

innerhalb einer POU, z.B. innerhalb des Programms PLC_PRG, kann diese gloable Variable nur unter Verwendung des Praefixes GVL angesprochen werden, bspw.

```
GVL.iVar:=5;
```

Der folgende unvollständige Aufruf der Variablen wird hingegen einen Fehler erzeugen:

```
iVar:=5;
```

Attribut Reflection

Das Pragma **{attribute 'reflection'}** wird Signaturen zugefügt. Aus Optimierungsgründen ist das Attribut zwingend an Bausteine zu vergeben, welche das. instance-path Attribut enthalten.

Syntax:

{attribute 'reflection’}

Beispiel:

siehe das Beispiel vom Attribut instance-path.

Attribut Symbol

Das Pragma **{attribute 'symbol'}** definiert, welche Variablen in der Symbolkonfiguration aufgenommen werden sollen, d.h. welche Variablen als "Symbole" in die Symbolliste exportiert werden sollen, die zum einen als XML-Datei im Projektverzeichnis wie auch als für den Anwender nicht sichtbare Datei auf dem Zielsystem für externen Zugriff bereitsteht, z.B. für Zugriff durch einen OPC-Server. Die mit diesem Symbol ausgestatteten Variablen werden auf die Steuerung geladen, selbst wenn sie im Editor der Symbolkonfiguration nicht explizit konfiguriert bzw. sichtbar sind.

Beachten Sie, dass die Symbolkonfiguration als Objekt unterhalb der betreffenden Applikation im Gerätebaum angelegt sein muss.

Syntax:

{attribute 'symbol' := 'none' | 'read' | 'write' | 'readwrite'}

Beachten Sie, dass nur Zugriff auf Symbole aus **Programmen** oder **Globale Variablenlisten** möglich ist. Um auf ein Symbol zuzugreifen, muss der Symbolname vollständig angegeben werden.

Die Pragma-Anweisung kann einzelnen Variablen hinzugefügt werden oder kollektiv allen Variablen, die in einem Programm deklariert sind::

- Um nur auf eine einzelne Variable zu wirken, muss das Pragma in der Zeile vor der Variablendeklaration platziert werden:
- Um für alle Variablen im Deklarationsteil eines Programms wirksam zu werden, muss das Pragma in der ersten Zeile des Deklarationseditors platziert werden. Auch in diesem Fall können dann noch Anweisungen für einzelne Variablen explizit in der jeweiligen Zeile gesetzt werden.

Welcher **Zugriff** auf ein Symbol möglich ist, wird durch den Parameter bestimmt. Mögliche Angaben: 'none', 'read', 'write' or 'readwrite'. Wenn kein Parameter angegeben wird, gilt der Standardwert 'readwrite'.

Beispiel:

Mit der folgenden Konfiguration werden die Variablen A und B mit Lese- und Schreibrecht exportiert. Variable D wird mit Leserecht exportiert.

```
{attribute 'symbol' := 'readwrite'}
PROGRAM PLC_PRG
VAR
A : INT;
B : INT;
{attribute 'symbol' := 'none'}
C : INT;
{attribute 'symbol' := 'read'}
D : INT;
END_VAR
```

Bedingte Pragmas

Die ExST (Extended ST) Programmiersprache unterstützt verschiedene bedingte Pragma-Anweisungen, die die Code-Generierung im Prä-Kompilierungs oder Komplierungsprozess (Übersetzen des Projekts) beeinflussen.

Ob ein Implementierungs-Code für die Übersetzung berücksichtigt wird, kann beispielsweise davon abhängig gemacht werden, ob ein bestimmter Datentyp oder eine bestimmte Variable deklariert sind, ob ein Datentyp oder eine Variable ein bestimmtes Attribut haben, ob eine Variable einen bestimmten Datentyp hat, ob ein bestimmter Baustein oder eine Task vorhanden bzw. Teil des Aufrufbaums sind, etc.

Hinweis: Es ist nicht möglich für eine POU oder eine GVL, die im POUs Fenster angelegt sind, ein "{define...}" zu verwenden, das in einer Applikation deklariert ist. "defines" in Applikationen wirken nur auf Schnittstellen, die unterhalb der Applikation hängen.

{define *identifier string*}	Während des Prä-Kompilierungsprozesses werden alle nachfolgenden Instanzen des Bezeichners *identifier* mit der angegebenen Zeichenfolge *string* ersetzt, wenn diese nicht leer ist (was erlaubt ist). Der Bezeichner bleibt definiert und gültig bis zum Ende seines Anwendungsbereichs oder bis er durch eine {undefine} Anweisung aufgehoben wird. Wird benötigt z.B. für eine "bedingte Übersetzung", s.u.
{undefine *identifier*} .	Die Präprozessor-Definition des Bezeichners *identifier* (durch {define}, s.o.) wird aufgehoben, der Bezeichner ist ab jetzt wieder "undefiniert". Wenn der angegebene Bezeichner gerade gar nicht definiert ist, wird das Pragma ignoriert.
{IF *expr*} ... {ELSIF *expr*} ... {ELSE} ... {END_IF}	Dies sind Pragmas für die **bedingte** Kompilierung. Die angegebenen Ausdrücke *expr* müssen zur Kompilierungszeit konstant sein; sie werden in der Reihenfolge ausgewertet, in der sie hier erscheinen, bis einer der Ausdrücke einen nicht-Null-Wert zeigt. Der mit der Anweisung verknüpfte Text wird vorverarbeitet und dann übersetzt; die anderen Zeile werden ignoriert. Die Reihenfolge der Abschnitte ist festgelegt; jedoch sind die *elsif* und *else* Abschnitte optional und *elsif* Abschnitte dürfen beliebig oft vorkommen. Innerhalb der Konstante *expr* können mehrere "bedingte Übersetzungs-Operatoren" verwendet werden, die unten beschrieben werden.

Bedingte Übersetzungs-Operatoren

Innerhalb des konstanten Ausdrucks **expr** eines bedingten Übersetzungs-Pragmas **{IF}** oder **{ELSIF}** - siehe oben - können ein oder mehrere Operatoren verwendet werden. Diese dürfen allerdings nicht selbst wiederum definiert {define} oder undefiniert {undefine} sein. Beachten Sie dass diese Ausdrücke genauso wie die Definition über ein {define} auch als "**Compiler-Definitionen**" in den Übersetzungs-Eigenschaften eines Objekts eingetragen werden können.

Folgende Operatoren werden unterstützt:

Operator	Wirkung und Beispiel

defined (*identifier*)	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn der Bezeichner *identifier* mit Hilfe einer {define}-Anweisung definiert wurde und danach nicht mit einer {undefine}-Anweisung wieder undefiniert wurde; ansonsten wird FALSE geliefert.
	Beispiel
	Voraussetzung: Es gibt zwei Applikationen App1 und App2. Variable "pdef1" ist durch eine {define}-Anweisung in App1 definiert, aber nicht in App2.
	{IF defined (pdef1)}
	(* dieser Code wird in App1 verarbeitet *)
	{info 'pdef1 defined'}
	hugo := hugo + SINT#1;
	{ELSE}
	(* dieser Code wird in App2 verarbeitet *)
	{info 'pdef1 not defined'}
	hugo := hugo - SINT#1;
	Hier ist zusätzlich ein Beispiel eines Meldungs-Pragmas enthalten: Nur die Information "pdef1 defined" wird im Meldungsfenster angezeigt, wenn die Applikation kompiliert wird, weil pdef1 tatsächlich definiert ist. Die Meldung "pdef1 not defined" wird ausgegeben, wenn pdef1 nicht definiert ist.

defined (variable:*variable*)	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn die Variable *variable* innerhalb des aktuellen Gültigkeitsbereichs deklariert ist; ansonsten wird FALSE geliefert. Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. Variable "hallo" ist in App1 deklariert, nicht aber in App2.Precondition: {IF defined (variable:g_bTest)} (* der folgende Code wird nur in App2 verarbeitet *) g_bTest := x > 300; {END_IF}

Operator	Wirkung und Beispiel
defined (type:*identifier*)	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn ein Datentyp mit Bezeichner *identifier* deklariert ist; ansonsten wird FALSE geliefert. Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. Datentyp "DUT" ist in App1 deklariert, nicht aber App2. {IF defined (type:DUT)} (* der folgende Code wird nur in App1 verarbeitet *) bDutDefined := TRUE; {END_IF}

defined (pou:*pou-name*)	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn ein Baustein oder eine Aktion mit dem Namen *pou-name* vorhanden ist; ansonsten wird FALSE geliefert.
	Beispiel
	Voraussetzung: Es gibt zwei Applikationen App1 und App2. Baustein "CheckBounds" ist in App1 vorhanden, nicht aber in App2.
	
	{IF defined (pou:CheckBounds)}
	(* der folgende Code wird nur in App1 verarbeitet *)
	arrTest[CheckBounds(0,i,10)] :=
	arrTest[CheckBounds(0,i,10)] + 1;
	{ELSE}
	(* der folgende Code wird nur in App2 verarbeitet *)
	arrTest[i] := arrTest[i]+1;
	{END_IF}

not yet implemented: defined (task:*identifier*)	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn eine Task mit dem Namen *identifier* definiert ist; ansonsten wird FALSE geliefert.
	Beispiel
	Voraussetzung: Es gibt zwei Applikationen App1 und App2. Task "PLC_PRG_Task" ist in App1 definiert, nicht aber in App2.
	
	{IF defined (task:PLC_PRG_Task)}
	(* der folgende Code wird nur in App1 verarbeitet *)
	erg := plc_prg.x;
	{ELSE}
	(* der folgende Code wird nur in App2 verarbeitet *)
	erg := prog.x;
	{END_IF}

Operator	Wirkung und Beispiel

not yet implemented: defined (resource:*identifier)*	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn ein Resourcen-Objekt mit dem Namen *identifier* für die Applikation vorhanden ist; ansonsten wird FALSE geliefert.
	Beispiel
	Voraussetzung: Es gibt zwei Applikationen App1 und App2. Ein Resourcen-Objekt "glob_var1" (global variable list) ist für App1 vorhanden, nicht aber für App2.
	
	{IF defined (resource:glob_var1)}
	(* der folgende Code wird nur in App1 verarbeitet *)
	gvar_x := gvar_x + ivar;
	{ELSE}
	(* der folgende Code wird nur in App2 verarbeitet *)
	x := x + ivar;
	{END_IF}

Operator	Wirkung und Beispiel
hasattribute (pou: pou-name, 'attribute')	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn das Attribut *attribute* in der ersten Zeile des Deklarationsteils des Bausteins pou-name angegeben ist; ansonsten wird FALSE geliefert. Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. Ein Funktion "fun1" ist in App1 und App2 definiert, aber in App1 ist ihr außerdem das Attribut 'vision' zugewiesen: Definition von fun1 in App1: {attribute 'vision'} FUNCTION fun1 : INT VAR_INPUT i : INT; END_VAR VAR END_VAR Definition von fun1 in App2: FUNCTION fun1 : INT VAR_INPUT i : INT; END_VAR VAR END_VAR Pragma-Anweisung: {IF hasattribute (pou: fun1, 'vision')} (* der folgende Code wird nur in App1 verarbeitet *) ergvar := fun1(ivar); {END_IF}

Operator	Wirkung und Beispiel
hasattribute (variable: *variable*, 'attribute')	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn der Variable *variable* das Attribut *attribute* mit Hilfe der {attribute} Anweisung in der Zeile vor der Variablendeklaration zugewiesen ist; ansonsten wird FALSE geliefert. Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. Variable g_globalInt wird in App1 und App2 verwendet, aber in App1 ist ihr zusätzlich das Attribut 'DoCount' zugewiesen: Deklaration von g_globalInt in App1 : VAR_GLOBAL {attribute 'DoCount'} g_globalInt : INT; g_multiType : STRING; END_VAR Deklaration von g_globalInt in App2 : VAR_GLOBAL g_globalInt : INT; g_multiType : STRING; END_VAR Pragma-Anweisung: {IF hasattribute (variable: g_globalInt, 'DoCount')} (* die folgende Code-Zeile wird nur in App1 ausgeführt, weil dort die Variable g_globalInt mit dem Attribut 'DoCount' definiert ist *) g_globalInt := g_globalInt + 1; {END_IF}

Operator	Wirkung und Beispiel
hastype (variable:*variable*, *type-spec*)	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn die Variable *variable* vom Datentyp type-spec ist; ansonsten wird FALSE geliefert. ANY ANY_DERIVED ANY_ELEMENTARY ANY_MAGNITUDE ANY_BIT ANY_STRING ANY_DATE ANY_NUM ANY_REAL ANY_INT LREAL REAL LINT DINT INT SINT ULINT UDINT UINT USINT TIME LWORD DWORD WORD BYTE BOOL STRING WSTRING DATE_AND_TIME DATE TIME_OF_DAY Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. Variable "g_multitype" ist in App1 mit Datentyp LREAL deklariert, in App2 mit Datentyp STRING: {IF (hastype (variable: g_multitype, LREAL))} (* die folgende Code-Zeile wird nur in App1 ausgeführt *) g_multitype := (0.9 + g_multitype) * 1.1; {ELSIF (hastype (variable: g_multitype, STRING))} (* die folgende Code-Zeile wird nur in App2 ausgeführt *) g_multitype := 'dies ist ein multitalent'; {END_IF}

Operator	Wirkung und Beispiel

hasvalue (*define-ident*, *char-string*)	Dieser Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn eine Variable mit Bezeichner *define-ident* definiert ist und den Wert *char-string* hat; ansonsten wird FALSE geliefert.
	Beispiel
	Voraussetzung: Es gibt zwei Applikationen App1 und App2. Variable "test" wird in den Applikationen App1 und App2 verwendet; in App1 erhält sie den Wert "1", in App2 Wert "2":
	{IF hasvalue(test,'1')}
	(* der folgende Code wird in App1 ausgeführt, weil dort Variable test den Wert 1 hat *)
	x := x + 1;
	{ELSIF hasvalue(test,'2')}
	(* der folgende Code wird in App1 ausgeführt, weil dort Variable test den Wert 2 hat *)
	x := x + 2;
	{END_IF}

NOT *operator*	Der Ausdruck erhält den Wert TRUE, wenn der Umkehrwert von Operator *operator* TRUE liefert *operator* kann einer der in der vorliegenden Tabelle beschriebenen Operatoren sein. Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. POU "PLC_PRG1" ist in App1 und App2 vorhanden, POU "CheckBounds" gibt es nur in App1: {IF defined (pou: PLC_PRG1) AND NOT (defined (pou: CheckBounds))} (* die folgende Code-Zeile wird nur in App2 ausgeführt *) bANDNotTest := TRUE; {END_IF}
operator AND *operator*	Der Ausdruck liefert TRUE, wenn die angegebenen Operatoren *operator* beide TRUE liefern. *operator* kann jeweils einer der in der vorliegenden Tabelle beschriebenen Operatoren sein. Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. POU "PLC_PRG1" ist in App1 und App2 vorhanden, POU "CheckBounds" nur in App1: {IF defined (pou: PLC_PRG1) AND (defined (pou: CheckBounds))} (* die folgende Code-Zeile wird nur in App1 ausgeführt, weil nur da sowohl "PLC_PRG1" als auch "CheckBounds" definiert sind. *) bORTest := TRUE; {END_IF}

Operator	Wirkung und Beispiel
operator OR *operator*	Der Ausdruck liefert TRUE, wenn beide angegebenen Operatoren *operator* TRUE liefern. *operator* kann jeweils einer der in der vorliegenden Tabelle beschriebenen Operatoren sein. Beispiel Voraussetzung: Es gibt zwei Applikationen App1 und App2. POU "PLC_PRG1" ist in App1 und App2 vorhanden, POU "CheckBounds" nur in App1: {IF defined (pou: PLC_PRG1) OR (defined (pou: CheckBounds))} (* die folgende Code-Zeile wird in App1 und App2 ausgeführt, weil beide wenigstens einen der POUS "PLC_PRG1" and "CheckBounds" enthalten *) bORTest := TRUE; {END_IF}
(operator)	Operatorklammern

16.1.27 Komponenten Auflisten Funktion

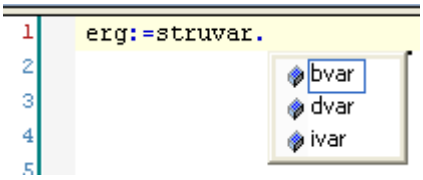
Die Texteingabe wird unter Berücksichtigung der Norm IEC 61131-3 unterstützt. Das Plug-In 'IEC 61131-3 Text Editor' bietet zusätzlich eine Art "Komponenten Auflisten"-Funktionalität, die im Projekt über die Texteditor-Optionen aktiviert oder deaktiviert werden kann.

"Komponenten Auflisten" erleichtert die Eingabe von gültigen Bezeichnern:

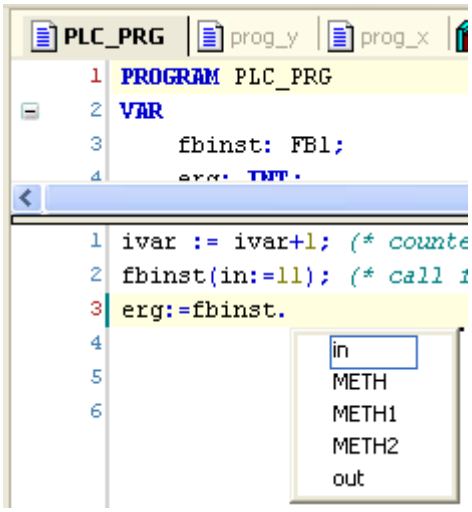
- Wenn an einer Stelle, an der ein globaler Bezeichner eingegeben werden kann, statt dessen nur ein Punkt "." eingegeben wird, erscheint eine Auswahlliste aller verfügbaren globalen Variablen. Eine Variable kann ausgewählt und mit hinter dem Punkt eingefügt werden.
- Wenn hinter einer Funktionsblockinstanz- oder Strukturvariablen ein Punkt eingegeben wird, bietet eine Auswahlliste alle Ein- und Ausgabevariablen des Funktionsblocks bzw. alle Strukturkomponenten an. Durch Doppelklick oder der <Eingabe>-Taste kann die in dieser Liste ausgewählte Variable hinter dem Punkt eingefügt werden.

Beispiele

"Komponenten Auflisten" bietet Strukturkomponenten an:



"Komponenten Auflisten" bietet Funktionsblockvariablen an:



- Wenn eine beliebige Zeichenfolge eingegeben und anschließend **<Strg> + <Leertaste>** gedrückt wird, erscheint eine Auswahlliste aller verfügbaren POUs und globalen Variablen. Das erste Element dieser Liste, das mit der vorher eingegebenen Zeichenfolge beginnt, wird es automatisch ausgewählt und kann über Doppelklick oder über die <Eingabe>-Taste im Editor eingefügt werden.

16.2 Datentypen

Beim Programmieren in »PLC Designer« können Standard Datentypen, selbst definierte Datentypen oder Instanzen von Funktionsblöcken verwendet werden. Jedem Bezeichner wird ein Datentyp zugeordnet, der festlegt, wieviel Speicherplatz reserviert wird und welche Werte dem Speicherinhalt entsprechen.

16.2.1 Standard Datentypen

Alle in der Norm IEC1131-3 beschriebenen Datentypen werden von »PLC Designer« unterstützt.

- BOOL
- Ganzzahlige Datentypen
- REAL / LREAL
- STRING
- WSTRING
- Zeitdatentypen

Beachten Sie, dass auch einige norm-erweiternde Datentypen verwendet werden können und dass Datentypen auch selbst definiert werden können (benutzerdefinierte Datentypen).

bool

Variablen vom Datentyp BOOL können die Wahrheitswerte TRUE (1) und FALSE (0) annehmen. Es werden 8 Bit Speicherplatz reserviert.

Siehe auch: BOOL Konstanten (Operanden)

Ganzzahlige Datentypen

Siehe im folgenden eine Liste der verfügbaren ganzzahligen Datentypen. Jeder dieser Typen deckt einen bestimmten Zahlenbereich ab, wobei die folgenden Bereichsgrenzen gelten:

Datentyp	Untergrenze	Obergrenze	Speicherplatz
BYTE	0	255	8 Bit
WORD	0	65535	16 Bit
DWORD	0	4294967295	32 Bit
LWORD	0	2^{64-1}	64 Bit
SINT	-128	127	8 Bit
USINT	0	255	8 Bit
INT	-32768	32767	16 Bit
UINT	0	65535	16 Bit
DINT	-2147483648	2147483647	32 Bit

Datentyp	Untergrenze	Obergrenze	Speicherplatz
UDINT	0	4294967295	32 Bit
LINT	-2^{63}	2^{63-1}	64 Bit
ULINT	0	2^{64-1}	64 Bit

Dadurch kann es passieren, dass bei der Typkonvertierung von größere auf kleinere Typen Information verloren geht.

Siehe auch: Zahlenkonstanten (Operanden)

REAL / LREAL

Die Datentypen REAL und LREAL sind sogenannte Gleitpunkttypen. Sie sind nötig bei Verwendung von rationalen Zahlen. Der reservierte Speicherplatz beträgt 32 Bit bei REAL und 64 Bit bei LREAL.

Zulässige Werte für REAL: 1.175494351e-38 bis 3.402823466e+38

Zulässige Werte für LREAL: 2.2250738585072014e-308 bis 1.7976931348623158e+308

Hinweis: Wenn ein REAL oder LREAL in SINT, USINT, INT, UINT, DINT, UDINT, LINT oder ULINT konvertiert wird und der Wert der REAL/LREAL-Zahl außerhalb des Wertebereichs des Integers liegt, erhält man ein undefiniertes Ergebnis das vom Zielsystem abhängt. Auch ein Ausnahmefehler ist dann möglich! Um zielsystemunabhängigen Code zu erhalten, müssen Wertebereichüberschreitungen über die Applikation abgefangen werden. Wenn die REAL/LREAL Zahl innerhalb des Bereichs liegt, arbeitet die Konvertierung auf allen Systemen gleich.

Siehe auch: REAL-/LREAL Konstanten (Operanden)

STRING

Eine Variable vom Datentyp STRING kann eine beliebige Zeichenkette aufnehmen. Die Größenangabe zur Speicherplatzreservierung bei der Deklaration bezieht sich auf Zeichen und kann in runden oder eckigen Klammern erfolgen. Ist keine Größe angegeben, so werden standardmäßig 80 Zeichen angenommen.

Die String-Länge ist grundsätzlich nicht begrenzt, allerdings können die String-Funktionen nur Längen von 1-255 verarbeiten! Wenn eine Variable mit einem String initialisiert wird, der zu lang ist für den Datentyp der Variablen, wird der String von hinten her entsprechend abgeschnitten.

Beispiel einer Stringdeklaration mit 35 Zeichen:

```
str:STRING(35):='This is a String';
```

Siehe auch:

- WSTRING
- STRING Konstanten (Operanden)

Zeitdatentypen

Die Datentypen **TIME**, **TIME_OF_DAY** (kurz **TOD**), **DATE** und **DATE_AND_TIME** (kurz **DT**) werden intern wie DWORD behandelt.

Bei TIME und TOD wird die Zeit in Millisekunden angegeben, wobei bei TOD ab 00:00 Uhr gerechnet wird.

Bei DATE und DT wird die Zeit in Sekunden angegeben, wobei ab dem 1. Januar 1970 um 00:00 Uhr gerechnet wird.

Beachten Sie in diesem Zusammenhang auch:

- LTIME, in Erweiterung der Norm IEC 61131-3 als 32-Bit Zeitdatentyp unterstützt.
- TIME Konstanten
- DATE Konstanten
- DATE_AND_TIME Konstanten
- TIME_OF_DAY Konstantens

16.2.2 Erweiterungen zur Norm IEC 61131-3

Normerweiternde Datentypen

Zusätzlich zu den Standard Datentypen werden auch folgende normerweiternde Datentypen in »PLC Designer« unterstützt:

- UNION
- ANY_TYPE
- LTIME
- WSTRING
- Referenzen
- Zeiger (Pointer)

union

In **Erweiterung der Norm IEC 61131-3** ist es möglich, "Unions" in benutzerdefinierten Datentypen zu deklarieren.

In einer Union haben alle Komponenten den gleichen Offset, d.h. sie belegen denselben Speicherplatz. In der folgenden Beispieldeklaration einer Union würde also eine Zuweisung auf name.a auch name.b betreffen:

Beispiel:

```
TYPE name: UNION
a : LREAL;
b : LINT;
END_UNION
END_TYPE
```

any_type

noch nicht verfügbar: In **Erweiterung der Norm IEC 61131-3** ist es möglich, Eingänge von Funktionen Methoden mit Datentyp ANY_TYPE zu deklarieren.

ltime

In Erweiterung der Norm IEC 61131-3 wird LTIME als Zeitbasis für hochauflösende Timer zu verwenden. LTIME hat die Größe von 64 Bit und eine Auflösung bis zum Nanosekunden-Bereich.

Syntax:

LTIME#<Zeitdeklaration>

Die Zeitdeklaration kann die Zeiteinheiten enthalten, die für die TIME Konstante gelten, und zusätzlich:

"us" : Mikrosekunden

"ns" : Nanosekunden

Beispiel:

LTIME1 := LTIME#1000d15h23m12s34ms2us44ns

Vergleiche dazu: TIME hat eine Größe von 32 Bit und eine Auflösung in Millisekunden.

wstring

Dieser String-Datentyp gilt in **Erweiterung zur Norm IEC 61131-3**.

Er unterscheidet sich vom Standard-String-Datentyp STRING (ASCII) dadurch, dass er in Unicode Format interpretiert wird.

Beispiel:

```
wstr:WSTRING:="This is a WString";
```

Siehe auch:

- STRING
- STRING Konstanten (Operanden)

Referenzen

Dieser String-Datentyp gilt in **Erweiterung zur Norm IEC 61131-3**.

Eine REFERENCE ist ein "Alias" für ein Objekt. Es kann über die Verwendung von Bezeichnern geschrieben oder gelesen werden. Der Unterschied zu einem Pointer ist der, dass der Wert, auf den gezeigt wird, direkt beeinflußt wird und dass die Zuweisung von Referenz und Wert fix ist. Die Adresse der Referenz muss über eine separate Zuweisungsoperation gesetzt werden. Ob eine Referenz auf einen gültigen Wert (d.h. ungleich 0) zeigt, kann mit Hilfe eines speziellen Operators geprüft werden, siehe unten.

Eine Referenz wird gemäß folgender Syntax deklariert:

Syntax

<Bezeichner> : REFERENCE TO <Datentyp>

Beispieldeklaration:

```
ref_int : REFERENCE TO INT;
a : INT;
b : INT;
```

ref_int kann nun als "Alias" für Variablen des Typs INT verwendet werden.

Anwendungsbeispiel:

```
ref_int REF= a;   (* ref_int zeigt nun auf a *)
ref_int := 12;    (* a hat nun den Wert 12 *)
b := ref_int * 2; (* b hat nun den Wert 24 *)
ref_int REF= b;   (* ref_int zeigt nun auf b *)
ref_int := a / 2; (* b hat nun den Wert 6 *)
ref_int REF= 0;    (* explizites Initialisieren der Referenz *)
```

Hinweis: Es ist nicht möglich, Referenzen folgendermaßen zu deklarieren:REFERENCE TO REFERENCE oder RRAY OF REFERENCE oder POINTER TO REFERENCE.

Hinweis: Mit Compiler-Version >= V3.3.0.0 werden Referenzen initialisiert (mit 0).

Prüfung auf gültige Referenzen

Operator "__ISVALIDREF" kann verwendet werden, um zu prüfen ob eine Referenz auf einen gültigen Wert weist, das heißt auf einen Wert ungleich 0.

Syntax:

<boolsche Variable> := __ISVALIDREF(<mit REFERENCE TO <datatype> deklarierter Kennzeichner);

<boolsche Variable> wird TRUE, wenn die Referenz auf einen gültigen Wert zeigt, andernfalls FALSE.

Beispiel:

Deklaration:

```
ivar : INT;
ref_int : REFERENCE TO INT;
ref_int0: REFERENCE TO INT;
testref: BOOL := FALSE;
```

Implementierung:

```
ivar := ivar +1;
ref_int REF= hugo;
ref_int0 REF= 0;
testref := __ISVALIDREF(ref_int);  (* wird TRUE, weil ref_int auf ivar zeigt, das einen Wert ungleich 0 hat *)
testref0 := __ISVALIDREF(ref_int0);   (* wird FALSE, weil ref_int auf 0 gesetzt ist *)
```

Zeiger / Pointer

In **Erweiterung zur Norm IEC 61131-3** wird die Verwendung von Zeigern (Pointer) unterstützt:

Pointer speichern die Adressen von Variablen, Programmen, Funktionsblöcken, Methoden und Funktionen, während ein Applikationsprogramm läuft. Ein Pointer kann auf jedes der genannten Objekte zeigen, sowie auf jeden Datentyp, sogar auf benutzerdefinierte Datentypen. Beachten Sie die Möglichkeit implizite Überwachungsfunktionen für Zeiger zu verwenden!

Syntax einer Pointer-Deklaration:

<Kennzeichner>: POINTER TO <Datentyp | Funktionsblock | Programm | Methode | Funktion>;

Einen Pointer zu dereferenzieren bedeutet, den Wert zu erhalten, der gerade auf der Adresse liegt, auf die der Pointer zeigt. Ein Pointer kann dereferenziert werden, indem der Inhaltsoperator "^" an den Pointer-Bezeichner angehängt wird, siehe z.B. "pt^" im unten gezeigten Beispiel.

Der Adress-Operator ADR kann verwendet werden, um einem Pointer die Adresse einer Variablen zuzuweisen.

Beispiel:

```
VAR
  pt:POINTER TO INT;  (* Deklaration von Pointer pt *)
  var_int1:INT := 5;  (* Deklaration der Variablen var_int1 und var_int2 *)
  var_int2:INT;
END_VAR
pt := ADR(var_int1); (* Adresse von varint1 wird Pointer pt zugewiesen *)
var_int2:= pt^;     (* Wert 5 von var_int1 wird durch Dereferenzieren von Pointer pt der Variablen var_int2 zugewiesen *)
```

Funktionszeiger

Im Unterschied zu »PLC Designer« V2.3 werden auch **Funktionszeiger** unterstützt, die den INDEXOF Operator ersetzen. Diese Zeiger können an externe Bibliotheken weitergereicht werden, aber es gibt **keine Möglichkeit, einen Funktionszeiger innerhalb** einer Applikation im Programmiersystem aufzurufen! Die Funktion des Laufzeitsystems zum Registrieren von Callback-Funktionen (Systembibliotheksfunktion) erwartet den Funktionszeiger, und je nachdem für welchen Callback man sich registrieren ließ, wird die betreffende Funktion dann vom LZS implizit aufgerufen (zum Beispiel bei Stop). Damit ein solcher **Systemaufruf möglich ist** (Laufzeitsystem), muss die entsprechende Objekteigenschaft (Kategorie "Übersetzen") für das Funktions-Objekt gesetzt werden.

Der ADR-Operator kann für Funktionen, Programme, Funktionsblöcke und Methoden verwendet werden. Da Funktionen ihren Wert nach einem Online Change ändern können, wird nicht die Adresse der Funktion, sondern die Adresse eines Pointers auf die Funktion ausgegeben. Diese Adresse gilt solange die Funktion auf dem Zielsystem existiert.

Indexzugriff auf Zeiger (Pointer)

In **Erweiterung zur Norm IEC 61131-3** ist der Indexzugriff "[]" auf Variablen vom Typ POINTER, STRING und WSTRING erlaubt.

- pint[i] ibt den Basisdatentyp zurück
- Indexzugriff auf Pointer erfolgt arithmetisch: Wenn der Indexzugriff bei einer Variablen vom Typ Pointer verwendet wird, wird der Offset durch pint[i] = (pint + i * SIZEOF(base type))^ errechnet. Der Indexzugriff bewirkt auch eine implizite Dereferenzierung des Pointers. Der resultierende Datentyp ist der Basisdatentyp des Pointers. Beachten Sie, dass pint[7] != (pint + 7)^!
- Wenn der Indexzugriff bei einer Variablen vom Typ STRING verwendet wird, erhält man das Zeichen an Offset Index-expr. Das Ergebnis ist vom Typ BYTE. str[i] gibt das i-te Zeichen des Strings als SINT (ASCII) zurück.
- Wenn der Indexzugriff bei einer Variablen vom Typ WSTRING verwendet wird, erhält man das Zeichen an Offset Index-expr. Das Ergebnis ist vom Typ WORD. wstr[i] gibt das i-te Zeichen des Strings als INT (Unicode) zurück.

Hinweis: Beachten Sie auch die Möglichkeit, Referenzen zu verwenden, die im Unterschied zu Zeigern einen Wert direkt beeinflussen.

Überwachungsfunktion für Zeiger

Zur Überwachung des Speicherzugriffs von Zeigern während der Laufzeit kann die implizite Überwachungsfunktion "CheckPointer" genutzt werden. Sie muss als Objekt "POUs für implizite Prüfungen" in die Applikation eingebunden werden mit Hilfe des Dialogs Objekt hinzufügen. Nachdem in dem entsprechendem Kontrollkästchen ein Häkchen gesetzt, eine Implementationssprache gewählt und diese Auswahl mit **Öffnen** bestätigt wurde, wird die Funktion CheckPointer im entsprechenden Editor geöffnet. In allen Sprachen ist der Deklarationsteil der Funktionen vorgegeben und darf nicht verändert werden! Als einzige Ausnahme dürfen lokale Variablen hinzugefügt werden. Im Unterschied zu anderen Überwachungsfunktionen existiert für die Implementierung von CheckPointer kein standardmäßiger Vorschlag; eine Implementierung ist vom Benutzer selbst vorzunehmen!

Die Funktion CheckPointer soll überprüfen, ob der übergebene Zeiger auf eine gültige Speicheradresse verweist und ob die Ausrichtung des referenzierten Speicherbereichs zum Typ der Variablen passt, auf die der Zeiger verweist. Sind beide Bedingungen erfüllt, so wird der Zeiger selbst zurückgegeben. Anderenfalls sollte CheckPointer eine angemessene Fehlerbehandlung durchführen.

Hinweis: Für den THIS Zeiger findet kein impliziter Aufruf der Überwachungsfunktion statt.

Schablone :

```
Deklarationsteil:
// Automatisch erzeugter Code: NICHT EDITIEREN
FUNCTION CheckPointer : POINTER TO BYTE
VAR_INPUT
ptToTest : POINTER TO BYTE;
iSize : DINT;
iGran : DINT;
bWrite: BOOL;
END_VAR
```

Implementierung (unvollständig!):

```
// Keine Standardimplementierung. Bitte fügen Sie Ihren Code hier ein.
CheckPointer := ptToTest;
```

Beim Aufruf werden der Funktion folgende Eingabeparameter übergeben:

- ptToTest: Zieladresse des Zeigers
- iSize: Größe der referenzierten Variable; der Datentyp von iSize muss kompatibel sein zu INT und den Größenbereich der Variablen abdecken
- iGran: Granularität der referenzierten Größe, d.h. der größte in der referenzierten Variablen enthaltene nicht-strukturierte Datentyp; der Datentyp von iGran muss kompatibel sein zu INT
- bWrite: Art des Zugriffs (TRUE=Schreibzugriff, FALSE=Lesezugriff); der Datentyp von bWrite muss BOOL sein

Bei positivem Ergebnis der Überprüfung wird der unveränderte Eingabezeiger zurückgegeben (ptToTest).

16.2.3 Benutzerdefinierte Datentypen

Außer den Standard Datentypen kann der Benutzer innerhalb eines Projekts auch eigene Datentypen definieren.

Diese Definitionen sind möglich, indem DUT (Data Unit Type) Objekte im POUs-Fenster angelegt werden, oder innerhalb der Deklaration eines Bausteins.

Beachten Sie bitte die Empfehlungen zur Namensvergabe für Objekte, um diese möglichst einheitlich zu gestalten.

Folgende benutzerdefinierten Datentypen können angelegt werden:

- ARRAY
- Structure
- Enumeration
- Subrange type
- Reference
- Pointer

Felder / Arrays

Es werden ein-, zwei-, und dreidimensionale Felder (Arrays) elementarer Datentypen unterstützt. Arrays können im Deklarationsteil eines Bausteins und in den globalen Variablenlisten definiert werden. Beachten Sie die Möglichkeit Funktionen zur Überprüfung der Feldgrenzen einzusetzen.

Syntax:

<Array_Name>:ARRAY [<ll1>..<ul1>,<ll2>..<ul2>] OF <elem. Type>

ll1, ll2, ll3 bezeichnen die Untergrenze einer Felddimension, ul1, ul2 und ul3 die Obergrenze. Diese Grenzwerte müssen ganzzahlige Werte sein.

Beispiel:

```
Card_game: ARRAY [1..13, 1..4] OF INT;
```

Initialisierung von Arrays:

Hinweis: Im Unterschied zu »PLC Designer« V2.3 müssen nun eckige Klammern um den/die Initialisierungswert(e) gesetzt werden !

Beispiel für die komplette Initialisierung eines Arrays:

```
arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];
arr2 : ARRAY [1..2,3..4] OF INT := [1,3(7)]; (* kurz für 1,7,7,7 *)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3];
    (* kurz für 0,0,4,4,4,4,2,3 *)
```

Beispiel für die Initialisierung eines Arrays einer Struktur:

Strukturdefinition:

```
TYPE STRUCT1
STRUCT
 p1:int;
 p2:int;
 p3:dword;
END_STRUCT
END_TYPE
```

Array-Initialisierung:

```
ARRAY[1..3] OF STRUCT1:= [(p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299),
(p1:=14,p2:=5,p3:=112)];
```

Beispiel für die teilweise Initialisierung eines Arrays:

```
arr1 : ARRAY [1..10] OF INT := [1,2];
```

Elemente, denen kein Wert explizit als Initialisierungswert zugewiesen wird, werden mit dem Default-Wert des Basisdatentyps initialisiert. Im obigen Beispiel werden die Elemente anarray[6] to anarray[10] also mit 0 initialisiert.

Zugriff auf Array-Komponenten:

In einem zweidimensionalen Array wird folgendermaßen auf die Komponenten zugegriffen:

<Array-Name>[Index1,Index2]

Beispiel:

```
Card_game [9,2]
```

Funktionen zur Überprüfung der Feldgrenzen

Um einen korrekten Zugriff auf Feldelemente zu gewährleisten muss der Applikation die Funktion CheckBounds zur Verfügung stehen. Sie wird als Objekt "POUs für implizite Prüfungen" in die Applikation eingebunden mit Hilfe des Dialogs Objekt hinzufügen. Nachdem in dem Kontrollkästchen vor CheckBounds ein Häkchen gesetzt, eine Implementierungssprache gewählt und diese Auswahl mit **Öffnen** bestätigt wurde, wird die Funktion CheckBounds im entsprechenden Editor geöffnet. In allen Sprachen ist der Deklarationsteil der Funktion vorgegeben und darf nicht verändert werden! Als einzige Ausnahme dürfen lokale Variablen hinzugefügt werden. Ein Vorschlag zur Implementierung der Funktion steht in ST zur Verfügung; dieser darf beliebig abgeändert werden.

Aufgabe dieser Überwachungsfunktion ist eine angemessene Behandlung von Verletzungen von Feldgrenzen (bspw. durch Setzen eines Errorflags oder durch Verändern des Feldindices). Der Aufruf der Funktion erfolgt implizit sobald einer Variablen vom Typ Array Werte zugewiesen werden.

Achtung: Um die Funktionalität der Überwachungsfunktion zu erhalten darf der Deklarationsteil der Funktion nicht abgeändert werden!

Beispiel für den Gebrauch der Funktion CheckBounds:

Standardmäßig ist die Funktion auf folgende Weise in ST implementiert:

Deklarationsteil:

```
// Automatisch erzeugter Code : NICHT EDITIEREN
FUNCTION CheckBounds : DINT
VAR_INPUT
index, lower, upper: DINT;
END_VAR
```

Implementierung:

```
// Automatisch erzeugter Code : Es handelt sich hierbei um einen
Implementierungsvorschlag.
IF  index < lower THEN
CheckBounds := lower;
ELSIF  index > upper THEN
CheckBounds := upper;
ELSE
CheckBounds := index;
END_IF
```

Beim Aufruf erhält die Funktion folgende Eingangsparameter:

- index: Index des Feldelements
- lower: Untergrenze der Felddimension
- upper: Obergrenze der Felddimension

Rückgabewert ist der Index des Feldelements, sofern sich dieser im gültigen Bereich befindet. Ansonsten wird je nach Verletzung des Grenzbereichs Ober- oder Untergrenze zurückgegeben.

Im untenstehendem Programm wird die definierte Obergrenze des Feldes a überschritten:

```
PROGRAM PLC_PRG
VAR
 a: ARRAY[0..7] OF BOOL;
 b: INT:=10;
END_VAR
a[b]:=TRUE;
```

In diesem Fall bewirkt der bei der Zuweisung implizit erfolgte Aufruf der Funktion CheckBounds, dass der Index 10 abgeändert wird in die Obergrenze des Feldbereichs von a. Damit wird der Wert TRUE dem Element a[7] zugewiesen. Auf diese Weise werden versuchte Feldzugriffe außerhalb des gültigen Feldbereichs korrigiert.

Strukturen

Strukturen werden als "DUT" (Data Type Unit) Objekte über den Dialog Objekt hinzufügen im Projekt angelegt.

Sie beginnen mit den Schlüsselwörtern TYPE und STRUCT und enden mit END_STRUCT und END_TYPE.

Hinweis: Im Unterschied zu »PLC Designer« V2.3 muss nach "TYPE" in der Struktur-Deklaration ein Doppelpunkt ":" folgen!

Die Syntax für Struktur-Deklarationen ist folgende:

```
TYPE <Strukturname>:
STRUCT
 <Variablendeklaration 1>
 .
 .
 <Variablendeklaration n>
END_STRUCT
END_TYPE
```

<Strukturname> ist ein Typ, der im gesamten Projekt erkannt wird und wie ein Standard Datentyp verwendet werden kann.

Verschachtelte Strukturen sind erlaubt. Die einzige Beschränkung ist, dass Variablen nicht Adressen zugewiesen werden dürfen (die AT-Deklaration ist nicht erlaubt!).

Beispiel für eine Strukturdefinition namens Polygonline:

```
TYPE Polygonline:
STRUCT
 Start:ARRAY [1..2] OF INT;
 Point1:ARRAY [1..2] OF INT;
 Point2:ARRAY [1..2] OF INT;
 Point3:ARRAY [1..2] OF INT;
 Point4:ARRAY [1..2] OF INT;
 End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

Initialisierung von Strukturen:

Beispiel:

```
Poly_1:polygonline := ( Start:=[3,3], Point1 =[5,2], Point2:=[7,3], Point3:=[8,5],
Point4:=[5,7], End := [3,5]);
```

Initialisierungen mit Variablen sind nicht erlaubt. Ein Beispiel für die Initialisierung eines Arrays einer Struktur finden Sie bei 'Felder (Arrays)'.

Zugriff auf Strukturkomponenten:

Der Zugriff auf eine Strukturkomponente erfolgt nach der Syntax:

<Strukturname>.<Komponentenname>

Für das obige Beispiel der Struktur 'Polygonline' kann also mit `Poly_1.Start`. auf die Komponente 'start' zugegriffen werden.

Aufzählungen / Enumerationen

Eine Enumeration ist ein benutzerdefinierter Datentyp, der sich aus einer Reihe von String-Konstanten zusammensetzt.

Aufzählungswerte (Enumerationswerte) sind global im Projekt bekannt, auch wenn sie innerhalb eines Bausteins deklariert sind.

Eine Enumeration wird als "DUT" (Data Unit Type) Objekt über den Dialog Objekt hinzufügen im Projekt angelegt.

Hinweis: In contrast to »PLC Designer« V2.3 a local enumeration declaration is no longer possible except within TYPE.

Syntax:

TYPE <Bezeichner>:(<Enum_0> ,<Enum_1>, ...,<Enum_n>) |<Basisdatentyp>;
END_TYPE

Eine Variable des Typs <Bezeichner> kann einen der Enumerationswerte <Enum_..> annehmen und wird mit dem ersten dieser Werte initialisiert. Die Werte sind kompatibel mit ganzen Zahlen, d.h. es können Operationen mit ihnen vorgenommen werden wie mit INTEGER-Variablen. Jedem Enumerationswert kann eine Zahl x zugewiesen werden. Wenn dies nicht explizit innerhalb der Deklaration erfolgt, erhält die erste Komponente 0 und die weiteren 1, 2 usw.. Wird die Zuweisung von Zahlenwerten innerhalb der Deklaration vorgenommen, muss darauf geachtet werden, dass es sich um eine aufsteigende Zahlenreihe innerhalb der Enumeration handelt.

Beispiel:

```
TYPE TRAFFIC_SIGNAL: (red, yellow, green:=10);  (* Die Initialwerte für die Farben sind red=0, yellow=1, green=10 *)
END_TYPE
TRAFFIC_SIGNAL1 : TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0; (* Der Wert des Ampelsignals ist "rot" *)
FOR i:= red TO green DO
 i := i + 1;
END_FOR;
```

Erweiterungen zur Norm IEC 61131-3:

1. Der Typname von Enumerationen kann (als Namensraum-Operator) verwendet werden, um den Zugriff auf eine Enumerationskonstante eindeutig zu machen. Somit ist es möglich, **die gleiche Konstante in verschiedenen Enumerationen** zu verwenden.

Beispiel:

Definition zweier Enumerationen:

```
TYPE COLORS_1: (red, blue);
END_TYPE
TYPE COLORS_2: (green, blue, yellow);
END_TYPE
```

Verwendung von Enumerationswert "blue" in einem Baustein:

Deklaration:

```
colorvar1 : COLORS_1;
colorvar2 : COLORS_2;
```

Implementation:

```
(* möglich: *)
colorvar1 := colors_1.blue;
colorvar2 := colors_2.blue;
(* nicht möglich: *)
colorvar1 := blue;
colorvar2 := blue;
```

2. Der Basisdatentyp der Enumeration - der standardmäßig INT ist - kann explizit definiert werden.

Beispiel:

Der Basisdatentyp für Enumeration BigEnum soll DINT sein:

```
TYPE BigEnum : (yellow, blue, green:=16#8000) DINT;
END_TYPE
```

Unterbereichstypen

Ein Unterbereichstyp ist ein benutzerdefinierter Datentyp , dessen Wertebereich nur eine Untermenge eines Basistypen umfaßt. Beachten Sie die Möglichkeit einer impliziten Bereichsüberwachung. Die Deklaration kann in einem DUT (Data Type Unit) Objekt erfolgen, eine Variablen kann aber auch direkt mit einem Unterbereichstypen deklariert werden:

Syntax für die Deklaration als Data Unit Type (DUT):

TYPE <Name> : <Inttype> (<ug>..<og>) END_TYPE;

<Name>	Muss ein gültiger IEC-Bezeichner sein
<Inttype>	Einer der Datentypen SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD).
<ug>	Konstante, die mit dem Basisdatentyp kompatibel sein muss und die die Untergrenze des Bereichstypen festlegt. Die Untergrenze selbst gehört zu diesem Bereich.
<og>	Konstante, die mit dem Basisdatentyp kompatibel sein muss und die die Obergrenze des Bereichstypen festlegt. Die Obergrenze selbst gehört zu diesem Basistypen.

Beispiele:

```
TYPE
 SubInt : INT (-4095..4095);
END_TYPE
```

Direkte Deklaration einer Variablen mit einem Unterbereichstypen:

```
VAR
 i : INT (-4095..4095);
 ui : UINT (0..10000);
END_VAR
```

Wird einem Unterbereichstypen ein Wert zugewiesen (in der Deklaration oder in der Implementation), der nicht in diesen Bereich fällt (z.B. i:=5000), wird eine Fehlermeldung ausgegeben.

Funktionen zur Bereichsüberwachung

Um die Bereichsgrenzen eines Unterbereichstypen während der Laufzeit zu überwachen muss in die Applikation die Funktion **CheckRangeSigned** bzw. **CheckRangeUnsigned eingebunden sein**. Sie werden als Objekt "POUs für implizite Prüfungen" in die Applikation eingebunden mit Hilfe des Dialogs Objekt hinzufügen. Nachdem in dem entsprechendem Kontrollkästchen ein Häkchen gesetzt, eine Implementierungssprache gewählt und diese Auswahl mit **Öffnen** bestätigt wurde, wird die Funktion CheckRangeSigned bzw. CheckRangeUnsigned im entsprechenden Editor geöffnet. In allen Sprachen ist der Deklarationsteil der Funktionen vorgegeben und darf nicht verändert werden! Als einzige Ausnahme dürfen lokale Variablen hinzugefügt werden. Ein Vorschlag zur Implementierung der Funktionen steht in ST zur Verfügung; dieser darf beliebig abgeändert werden.

Aufgabe dieser Überwachungsfunktionen ist eine angemessene Behandlung von Verletzungen der Bereichsgrenzen (bspw. durch Setzen eines Errorflags oder durch Verändern eines Wertes). Der Aufruf der Funktionen erfolgt implizit bei der Zuweisung eines Wertes an eine Variable vom Unterbereichstyp.

Achtung: Um die Funktionalität der Überwachungsfunktion zu erhalten darf der Deklarationsteil der Funktion nicht abgeändert werden!

Beispiel:

Wird einer Variablen eines vorzeichenbehafteten Unterbereichstyps ein Wert zugewiesen, so bedingt dies einen automatischen Aufruf der Funktion CheckRangeSigned. Die Funktion, welche den Zuweisungswert auf den bei der Variablendeklaration festgesetzten Unterbereich beschränkt, ist standardmäßig wie folgt in ST implementiert:

Deklarationsteil:

```
// Automatisch erzeugter Code : NICHT EDITIEREN
FUNCTION CheckRangeSigned : DINT
VAR_INPUT
value, lower, upper: DINT;
END_VAR
```

Implementierung:

```
// Automatisch erzeugter Code : Es handelt sich hierbei um einen Implementierungsvorschlag.
IF (value < lower) THEN
CheckRangeSigned := lower;
ELSIF(value > upper) THEN
CheckRangeSigned := upper;
ELSE
CheckRangeSigned := value;
END_IF
```

Beim Aufruf werden der Funktion folgende Eingangsparameter übergeben:

- value: Wert der Variablen vom Unterbereichstyp zugewiesen werden soll
- lower: Bereichsuntergrenze
- upper: Bereichsobergrenze

Rückgabewert ist der Zuweisungswert selbst, sofern sich dieser im gültigen Bereich befindet. Ansonsten wird je nach Verletzung des Unterbereichs seine Ober- oder Untergrenze zurückgegeben.

Die Zuweisung i:=10*y wird nun implizit ersetzt durch

```
i := CheckRangeSigned(10*y, -4095, 4095);
```

Hat y beispielsweise den Wert "1000", so wird der Variablen i nicht wie im ursprünglichem Code vorgesehen der Wert "10*1000=10000" zugewiesen, sondern der Wert der Bereichsobergrenze, also "4095".

Dasselbe gilt für die Funktion CheckRangeUnsigned.

Hinweis: Steht keine der Funktionen CheckRangeSigned und CheckRangeUnsigned zur Verfügung, so erfolgt während der Laufzeit keine Überprüfung des Unterbereichs bei entsprechenden Variablen! In diesem Fall kann einer Variablen eines Unterbereichtyps von INT jeder Wert zwischen –32768 und 32767 zugewiesen werden!

Attention: Bei Einbindung der Funktionen CheckRangeSigned und CheckRangeUnsigned können Endlosschleifen entstehen. Dies ist beispielsweise der Fall, wenn die Zählvariable einer FOR-Schleife vom Unterbereichstyp ist und der Zählbereich der Schleife den definierten Unterbereich verlässt !

Beispiel einer Endlosschleife:

```
VAR
 ui : UINT (0..10000);
END_VAR

FOR ui:=0 TO 10000 DO
...
END_FOR
```

Die FOR Schleife wird niemals verlassen, da die Überwachungsfunktion CheckRangeSigned verhindert, dass ui auf einen Wert größer als 10000 gesetzt wird.

Referenzen

Dieser String-Datentyp gilt in **Erweiterung zur Norm IEC 61131-3**.

Eine REFERENCE ist ein "Alias" für ein Objekt. Es kann über die Verwendung von Bezeichnern geschrieben oder gelesen werden. Der Unterschied zu einem Pointer ist der, dass der Wert, auf den gezeigt wird, direkt beeinflußt wird und dass die Zuweisung von Referenz und Wert fix ist. Die Adresse der Referenz muss über eine separate Zuweisungsoperation gesetzt werden. Ob eine Referenz auf einen gültigen Wert (d.h. ungleich 0) zeigt, kann mit Hilfe eines speziellen Operators geprüft werden, siehe unten.

Eine Referenz wird gemäß folgender Syntax deklariert:

Syntax

<Bezeichner> : REFERENCE TO <Datentyp>

Beispieldeklaration:

```
ref_int : REFERENCE TO INT;
a : INT;
b : INT;
```

ref_int kann nun als "Alias" für Variablen des Typs INT verwendet werden.

Anwendungsbeispiel:

```
ref_int REF= a;   (* ref_int zeigt nun auf a *)
ref_int := 12;    (* a hat nun den Wert 12 *)
b := ref_int * 2; (* b hat nun den Wert 24 *)
ref_int REF= b;   (* ref_int zeigt nun auf b *)
ref_int := a / 2; (* b hat nun den Wert 6 *)
ref_int REF= 0;    (* explizites Initialisieren der Referenz *)
```

Hinweis: Es ist nicht möglich, Referenzen folgendermaßen zu deklarieren:REFERENCE TO REFERENCE oder RRAY OF REFERENCE oder POINTER TO REFERENCE.

Hinweis: Mit Compiler-Version >= V3.3.0.0 werden Referenzen initialisiert (mit 0).

Prüfung auf gültige Referenzen

Operator "__ISVALIDREF" kann verwendet werden, um zu prüfen ob eine Referenz auf einen gültigen Wert weist, das heißt auf einen Wert ungleich 0.

Syntax:

<boolsche Variable> := __ISVALIDREF(<mit REFERENCE TO <datatype> deklarierter Kennzeichner);

<boolsche Variable> wird TRUE, wenn die Referenz auf einen gültigen Wert zeigt, andernfalls FALSE.

Beispiel:

Deklaration:

```
ivar : INT;
ref_int : REFERENCE TO INT;
ref_int0: REFERENCE TO INT;
testref: BOOL := FALSE;
```

Implementierung:

```
ivar := ivar +1;
ref_int REF= hugo;
ref_int0 REF= 0;
testref := __ISVALIDREF(ref_int);  (* wird TRUE, weil ref_int auf ivar zeigt, das einen Wert ungleich 0 hat *)
testref0 := __ISVALIDREF(ref_int0);   (* wird FALSE, weil ref_int auf 0 gesetzt ist *)
```

Zeiger / Pointer

In **Erweiterung zur Norm IEC 61131-3** wird die Verwendung von Zeigern (Pointer) unterstützt:

Pointer speichern die Adressen von Variablen, Programmen, Funktionsblöcken, Methoden und Funktionen, während ein Applikationsprogramm läuft. Ein Pointer kann auf jedes der genannten Objekte zeigen, sowie auf jeden Datentyp, sogar auf benutzerdefinierte Datentypen. Beachten Sie die Möglichkeit implizite Überwachungsfunktionen für Zeiger zu verwenden!

Syntax einer Pointer-Deklaration:

<Kennzeichner>: POINTER TO <Datentyp | Funktionsblock | Programm | Methode | Funktion>;

Einen Pointer zu dereferenzieren bedeutet, den Wert zu erhalten, der gerade auf der Adresse liegt, auf die der Pointer zeigt. Ein Pointer kann dereferenziert werden, indem der Inhaltsoperator "^" an den Pointer-Bezeichner angehängt wird, siehe z.B. "pt^" im unten gezeigten Beispiel.

Der Adress-Operator ADR kann verwendet werden, um einem Pointer die Adresse einer Variablen zuzuweisen.

Beispiel:

```
VAR
  pt:POINTER TO INT;  (* Deklaration von Pointer pt *)
  var_int1:INT := 5;  (* Deklaration der Variablen var_int1 und var_int2 *)
  var_int2:INT;
END_VAR
pt := ADR(var_int1); (* Adresse von varint1 wird Pointer pt zugewiesen *)
var_int2:= pt^;     (* Wert 5 von var_int1 wird durch Dereferenzieren von Pointer pt der Variablen var_int2 zugewiesen *)
```

Funktionszeiger

Im Unterschied zu »PLC Designer« V2.3 werden auch **Funktionszeiger** unterstützt, die den INDEXOF Operator ersetzen. Diese Zeiger können an externe Bibliotheken weitergereicht werden, aber es gibt **keine Möglichkeit, einen Funktionszeiger innerhalb** einer Applikation im Programmiersystem aufzurufen! Die Funktion des Laufzeitsystems zum Registrieren von Callback-Funktionen (Systembibliotheksfunktion) erwartet den Funktionszeiger, und je nachdem für welchen Callback man sich registrieren ließ, wird die betreffende Funktion dann vom LZS implizit aufgerufen (zum Beispiel bei Stop). Damit ein solcher **Systemaufruf möglich ist** (Laufzeitsystem), muss die entsprechende Objekteigenschaft (Kategorie "Übersetzen") für das Funktions-Objekt gesetzt werden.

Der ADR-Operator kann für Funktionen, Programme, Funktionsblöcke und Methoden verwendet werden. Da Funktionen ihren Wert nach einem Online Change ändern können, wird nicht die Adresse der Funktion, sondern die Adresse eines Pointers auf die Funktion ausgegeben. Diese Adresse gilt solange die Funktion auf dem Zielsystem existiert.

Indexzugriff auf Zeiger (Pointer)

In **Erweiterung zur Norm IEC 61131-3** ist der Indexzugriff "[]" auf Variablen vom Typ POINTER, STRING und WSTRING erlaubt.

- pint[i] ibt den Basisdatentyp zurück
- Indexzugriff auf Pointer erfolgt arithmetisch: Wenn der Indexzugriff bei einer Variablen vom Typ Pointer verwendet wird, wird der Offset durch pint[i] = (pint + i * SIZEOF(base type))^ errechnet. Der Indexzugriff bewirkt auch eine implizite Dereferenzierung des Pointers. Der resultierende Datentyp ist der Basisdatentyp des Pointers. Beachten Sie, dass pint[7] != (pint + 7)^!
- Wenn der Indexzugriff bei einer Variablen vom Typ STRING verwendet wird, erhält man das Zeichen an Offset Index-expr. Das Ergebnis ist vom Typ BYTE. str[i] gibt das i-te Zeichen des Strings als SINT (ASCII) zurück.
- Wenn der Indexzugriff bei einer Variablen vom Typ WSTRING verwendet wird, erhält man das Zeichen an Offset Index-expr. Das Ergebnis ist vom Typ WORD. wstr[i] gibt das i-te Zeichen des Strings als INT (Unicode) zurück.

Hinweis: Beachten Sie auch die Möglichkeit, Referenzen zu verwenden, die im Unterschied zu Zeigern einen Wert direkt beeinflussen.

Überwachungsfunktion für Zeiger

Zur Überwachung des Speicherzugriffs von Zeigern während der Laufzeit kann die implizite Überwachungsfunktion "CheckPointer" genutzt werden. Sie muss als Objekt "POUs für implizite Prüfungen" in die Applikation eingebunden werden mit Hilfe des Dialogs Objekt hinzufügen. Nachdem in dem entsprechendem Kontrollkästchen ein Häkchen gesetzt, eine Implementationssprache gewählt und diese Auswahl mit **Öffnen** bestätigt wurde, wird die Funktion CheckPointer im entsprechenden Editor geöffnet. In allen Sprachen ist der Deklarationsteil der Funktionen vorgegeben und darf nicht verändert werden! Als einzige Ausnahme dürfen lokale Variablen hinzugefügt werden. Im Unterschied zu anderen Überwachungsfunktionen existiert für die Implementierung von CheckPointer kein standardmäßiger Vorschlag; eine Implementierung ist vom Benutzer selbst vorzunehmen!

Die Funktion CheckPointer soll überprüfen, ob der übergebene Zeiger auf eine gültige Speicheradresse verweist und ob die Ausrichtung des referenzierten Speicherbereichs zum Typ der Variablen passt, auf die der Zeiger verweist. Sind beide Bedingungen erfüllt, so wird der Zeiger selbst zurückgegeben. Anderenfalls sollte CheckPointer eine angemessene Fehlerbehandlung durchführen.

Hinweis: Für den THIS Zeiger findet kein impliziter Aufruf der Überwachungsfunktion statt.

Schablone :

```
Deklarationsteil:
// Automatisch erzeugter Code: NICHT EDITIEREN
FUNCTION CheckPointer : POINTER TO BYTE
VAR_INPUT
ptToTest : POINTER TO BYTE;
iSize : DINT;
iGran : DINT;
bWrite: BOOL;
END_VAR
```

Implementierung (unvollständig!):

```
// Keine Standardimplementierung. Bitte fügen Sie Ihren Code hier ein.
CheckPointer := ptToTest;
```

Beim Aufruf werden der Funktion folgende Eingabeparameter übergeben:

- ptToTest: Zieladresse des Zeigers
- iSize: Größe der referenzierten Variable; der Datentyp von iSize muss kompatibel sein zu INT und den Größenbereich der Variablen abdecken
- iGran: Granularität der referenzierten Größe, d.h. der größte in der referenzierten Variablen enthaltene nicht-strukturierte Datentyp; der Datentyp von iGran muss kompatibel sein zu INT
- bWrite: Art des Zugriffs (TRUE=Schreibzugriff, FALSE=Lesezugriff); der Datentyp von bWrite muss BOOL sein

Bei positivem Ergebnis der Überprüfung wird der unveränderte Eingabezeiger zurückgegeben (ptToTest).

16.3 Operatoren

16.3.1 IEC Operatoren und norm-erweiternde Funktionen

»PLC Designer« V3.x unterstützt alle IEC Operatoren. Im Unterschied zu den Standardfunktionen sind diese Operatoren implizit überall im Projekt bekannt.

Neben den IEC Operatoren werden auch **folgende Operatoren unterstützt, die nicht durch die IEC-Norm beschrieben sind**: ANDN, ORN, XORN, INDEXOF und SIZEOF (siehe Arithmetische Operatoren), ADR, BITADR und Inhaltsoperator (siehe Addressoperatoren), einige "Namensraum-Operatoren".

Operatoren werden in einem Baustein wie Funktionen benutzt.

Achtung: Bei Operationen mit Gleitkomma-Datentypen ist das Rechenergebnis abhängig von der verwendeten Zielsystem-Hardware !

Sehen Sie folgende Kategorien von Operatoren:

- Zuweisungsoperatoren: :=, MOVE
- Arithmetische Operatoren
- Bitstring Operatoren
- Bit-Shift Operatoren
- Auswahl-Operatoren
- Vergleichsoperatoren
- Adressoperatoren
- Aufrufoperator
- Typkonvertierungen
- Numerische Funktionen
- IEC-erweiternde Operatoren
- IEC-erweiternde "Namensraum-Operatoren"

16.3.2 Arithmetische Operatoren

Die folgenden Operatoren, beschrieben durch die Norm IEC 61131-3, werden unterstützt:

ADD, MUL, SUB, DIV, MOD, MOVE

Folgende Operatoren erweitern die Norm:

INDEXOF, SIZEOF

ADD

IEC Operator: Addition von Variablen.

Erlaubte Typen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL and LREAL.

Zwei TIME Variablen können addiert werden, das Ergebnis kann auch von einem andere Zeitdatentyp sein
(z.B.., t#45s + t#50s = t#1m35s).

Beispiel in AWL:

```
LD    7
ADD   2
ADD   4
ADD   7
ST    iVar
```

Beispiel in ST:

```
var1 := 7+2+4+7;
```

Beispiel in FUP:

MUL

IEC Operator: Multiplikation von Variablen.

Erlaubte Typen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL and LREAL.

Beispiel in AWL:

```
LD    7
MUL   2,
      4,
      7
ST    Var1
```

Beispiel in ST:

```
var1 := 7*2*4*7;
```

Beispiel in FUP:

SUB

IEC Operator: Subtraktion einer Variablen von einer anderen.

Erlaubte Typen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL and LREAL.

Eine TIME Variable kann auch von einer Variable eines anderen TIME-Typs subtrahiert werden, wobei das Ergebnis in einer Variable eines dritten TIME-Typs steht. Beachten Sie aber, dass negative TIME Werte undefiniert sind.

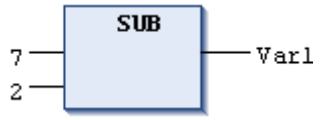
Beispiel in AWL:

```
LD      7
SUB     2
ST      Var1
```

Beispiel in ST:

```
var1 := 7-2;
```

Beispiel in FUP:



DIV

IEC Operator: Division einer Variablen durch eine andere.

Erlaubte Typen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL and LREAL.

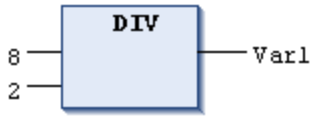
Beispiel in AWL: (Result in Var1 is 4)

```
LD      8
DIV     2
ST      Var1
```

Beispiel in ST:

```
var1 := 8/2;
```

Beispiel in FUP:



Achtung: Das Verhalten bei einer Division durch Null kann zielsystemabhängig unterschiedlich sein!

Funktionen zur Überprüfung

Um den Wert eines Divisors zu überwachen, um also ein Teilen durch 0 zu vermeiden können die Funktionen **CheckDivInt**, **CheckDivLint**, **CheckDivReal** und **CheckDivLReal** verwendet werden. Nach ihrer Einbindung in die Applikation geht jeder im zugehörigen Code auftretenden Division automatisch ihr Aufruf voraus. Zur Einbindung in die Applikation benutzen Sie Objekt hinzufügen: hierin wählen Sie das Objekt "POUs for implicit checks", setzen Sie ein Häkchen in dem entsprechenden Kontrollkästchen und wählen eine Implementierungssprache. Nach Bestätigung der Auswahl mit **Öffnen** wird die markierte Funktion in dem zur Sprachwahl passenden Editor geöffnet. Unabhängig von der Sprachwahl ist der Deklarationsteil der Funktionen fest vorgegeben und darf nicht verändert werden. Als einzige Ausnahme dürfen lokale Variablen zugefügt werden. Ein Vorschlag zur Implementierung der Funktionen steht in ST zur Verfügung; dieser darf beliebig modifiziert werden.

Sehen Sie folgendes Beispiel für eine Implementierung der Funktion CheckDivReal:

Standardmäßige Implementierung der Funktion CheckDivReal:

Deklarationsteil:

```
// Automatisch erzeugter Code : NICHT EDITIEREN
FUNCTION CheckDivReal : REAL
VAR_INPUT
 divisor:REAL;
END_VAR
```

Implementation part:

```
// Automatisch erzeugter Code: Es handelt sich hierbei um einen
Implementierungsvorschlag.
IF divisor = 0 THEN
 CheckDivReal:=1;
ELSE
 CheckDivReal:=divisor;
END_IF;
```

Der Operator DIV verwendet die Ausgabe der Funktion CheckDivReal als Divisor. In untenstehendem Beipielprogramm wird eine Division durch 0 verhindert, dadurch dass der implizit mit "0" initiierte Wert des Divisors "d" vor Ausführung der Teilung von der Funktion CheckDivReal in "1" geändert wird. Somit lautet das Ergebnis der Division 799.

```
PROGRAM PLC_PRG
VAR
 erg:REAL;
 v1:REAL:=799;
 d:REAL;
END_VAR
erg:= v1 / d;
```

ubMOD

IEC Operator: Modulo Division einer Variablen durch eine andere.

Erlaubte Typen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL and LREAL.

Als Ergebnis liefert diese Funktion den ganzzahligen Rest der Division.

Beispiel in AWL: (Ergebnis in Var1 ist 1)

```
LD    9
MOD   2
ST    Var1
```

Beispielin ST:

```
var1 := 9 MOD 2;
```

Beispielin FUP:

MOVE

IEC Operator: Zuweisung einer Variablen auf eine andere Variable eines entsprechenden Typs.

Dadurch, dass MOVE in den CFC-, FUP- und KOP-Editoren als Baustein verfügbar ist, kann dort die EN/EN0-Funktionalität auch auf eine Variablenzuweisung angewendet werden.

Beispiel in CFC in Verbindung mit der EN/ENO Funktion:

Nur wenn en_i TRUE ist, wird der Wert der Variablen var1 Variable var2 zugewiesen.

Beispiel in AWL: (Ergebnis: var2 erhält den Wert von var1)

```
LD      var1
MOVE
ST      var2
```

(! dies entspricht:

```
LD      var1
ST      var2
```

)

Beispiel in ST:

```
ivar2 := MOVE(ivar1);
```

(! dies entspricht: ivar2 := ivar1;)

SIZEOF

Dieser arithmetische Operator ist nicht von der Norm IEC61131-3 vorgeschrieben.

Er kann verwendet werden, um die Anzahl Bytes zu bestimmen, die von der angegebenen Variablen x benötigt werden

Der SIZEOF Operator liefert immer einen vorzeichenlosen Wert. Der Typ der Rückgabevariablen paßt sich der gefundenen Größe von Variable x an.

Rückgabewert von SIZEOF(x)	Datentyp der Konstanten, die implizit für die gefundene Größe verwendet wird
0 <= Größe von x < 256	USINT
256 <= Größe von x < 65536	UINT
65536 <= Größe von x < 4294967296	UDINT
4294967296 <= Größe von x	ULINT

Beispiel in ST:

```
var1 := SIZEOF(arr1);  (* d.h.:  var1:=USINT#10; *)
```

Beispiel in AWL: (Ergebnis ist 10)

```
arr1:ARRAY[0..4] OF INT;
Var1:INT;
LD      arr1
SIZEOF
ST      Var1
```

INDEXOF

Dieser arithmetische Operator ist nicht von der Norm IEC61131-3 vorgeschrieben.

Mit dieser Funktion kann der interne Index eines Bausteins ermittelt werden.

Beispiel in ST:

```
var1 := INDEXOF(POU2);
```

16.3.3 Bitstring Operatoren

De folgenden Bitstring Operatoren, beschrieben durch die Norm IEC 61131-3, werden unterstützt:

AND, OR, XOR, NOT

noch nicht verfübar: Die folgenden Bitstring-Operatoren werden in Erweiterung der Norm unterstützt:

ANDN, ORN, XORN

Bitstring Operatoren vergleichen die Bits von zwei oder mehreren Operanden.

AND

IEC Bitstring Operator: Bitweise AND von Bit-Operanden. Wenn die Eingangs-Bits jeweils 1 sind, ist das Ausgangs-Bit 1, ansonsten 0.

Erlaubte Typen: BOOL, BYTE, WORD, DWORD, LWORD.

Beispiel in AWL: (Result in Var1 is 2#1000_0010)

```
Var1:BYTE;
LD      2#1001_0011
AND     2#1000_1010
ST      var1
```

Beispiel in ST:

```
var1 := 2#1001_0011 AND 2#1000_1010
```

Beispiel in FUP:

ANDN

Noch nicht verfügbar.

Dieser IEC Bitstring Operator wird nicht von der Norm IEC61131-3 beschrieben.

OR

IEC Bitstring Operator: Bitwise OR of bit operands. Wenn mindestens einer der Eingangs-Bits 1 is, ist das Ausgangs-Bit 1, ansonsten 0.

Erlaubte Typen: BOOL, BYTE, WORD, DWORD, LWORD.

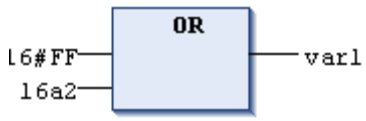
Beispiel in AWL: (Ergebnis in var1 ist 2#1001_1011)

```
var1 :BYTE;
LD    16#FF
OR    16a2
ST    var1
```

Beispiel in ST:

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

Beispiel in FUP:



ORN

Noch nicht verfügbar.

Dieser IEC Bitstring Operator wird nicht von der Norm IEC61131-3 beschrieben.

XOR

IEC Bitstring Operator: Bitweise XOR-Operation von Bit-Operanden. Wenn nur einer der beiden Eingangs-Bits den Wert 1 liefert, wird das Ausgangs-Bit 1; wenn beide Eingänge 1 oder beide 0 sind, wird der Ausgang 0.

Erlaubte Typen: BOOL, BYTE, WORD, DWORD, LWORD.

Hinweis: Beachten Sie das Verhalten des XOR-Bausteins in erweiterter Form, also bei mehr als 2 Eingängen: Die Eingänge werden paarweise geprüft und die jeweiligen Ergebnisse dann wiederum gegeneinander verglichen (entspricht der Norm, jedoch nicht unbedingt der Erwartung).

Beispiel in AWL: (Ergebnis ist 2#0001_1001)

```
Var1 :BYTE;
LD    2#1001_0011
XOR   2#1000_1010
ST    var1
```

Beispiel in ST:

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

Beispiel in FUP:

XORN

Noch nicht verfügbar.

Dieser IEC Bitstring Operator wird nicht von der Norm IEC61131-3 beschrieben.

NOT

IEC Bitstring Operator: Bitweises NOT eines Bit Operanden. Das Ausgangs-Bit ist 1, wenn das entsprechende Eingangsbit 0 ist und umgekehrt.

Erlaubte Typen: BOOL, BYTE, WORD, DWORD, LWORD.

Beispiel in AWL: (Result in Var1 is 2#0110_1100)

```
Var1 :BYTE;
LD    2#1001_0011
NOT
ST    var1
```

Beispiel in ST:

```
Var1 := NOT 2#1001_0011
```

Beispiel in FUP:

16.3.4 Bit-Shift Operatoren

Die folgenden Bit-Shift Operatoren, beschrieben durch die Norm IEC 61131-3, werden unterstützt::

SHL, SHR, ROL, ROR

SHL

IEC Operator: Bitweises Verschieben eines Operanden nach links (Shift).

```
erg:= SHL (in, n)
```

in: Operand, der nach links verschoben werden soll.

n: Anzahl der Bits, um die *in* nach links verschoben wird.

Wenn *n* die Datentypbreite überschreitet, werden BYTE, WORD, DWORD and LWORD Operanden mit Nullen aufgefüllt während Operanden von vorzeichenbehafteten Typen, wie z.B. INT, arithmetisch verschoben werden, d.h. sie werden mit dem Wert des obersten Bits aufgefüllt.

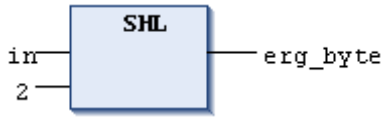
Hinweis: Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung. dass sie unterschiedliche Ergebnisse für erg_byte und erg_word erhalten, abhängig vom Datentyp der Eingangsvariablen (BYTE oder WORD), obwohl die Werte der Eingangsvariablen in_byte und in_word gleich sind.

Beispiel in ST:

```
PROGRAM shl_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
erg_byte:=SHL(in_byte,n); (* Ergebnis ist 16#14 *)
erg_word:=SHL(in_word,n); (* Ergebnis ist 16#011 *)
```

Beispiel in FUP:



Beispiel in AWL:

```
LD      in_byte
SHL     2
ST      erg_byte
```

SHR

IEC Operator: Bitweises Verschieben eines Operanden nach rechts (Shift).

```
erg:= SHR (in, n)
```

in: Operand, der nach rechts verschoben werden soll.

n: Anzahl der Bits, um die *in* nach rechts verschoben wird.

Wenn *n* die Datentypbreite überschreitet, werden BYTE, WORD, DWORD and LWORD Operanden mit Nullen aufgefüllt während Operanden von vorzeichenbehafteten Typen, wie z.B. INT, arithmetisch verschoben werden, d.h. sie werden mit dem Wert des obersten Bits aufgefüllt.

Hinweis: Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung. dass sie unterschiedliche Ergebnisse für erg_byte und erg_word erhalten, abhängig vom Datentyp der Eingangsvariablen (BYTE oder WORD), obwohl die Werte der Eingangsvariablen in_byte und in_word gleich sind.

Beispiel in ST:

```
PROGRAM shr_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
erg_byte:=SHR(in_byte,n); (* Ergebnis ist 11 *)
erg_word:=SHR(in_word,n); (* Ergebnis ist 0011 *)
```

Beispiel in FUP:

Beispiel in AWL:

```
LD    in_byte
SHR   2
ST    erg_byte
```

ROL

IEC Operator: Bitweise Rotation eines Operanden nach links.

```
erg:= ROL (in, n)
```

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD.

in wird *n* mal um 1 Bit nach links verschoben und gleichzeitig das Bit mit der äußersten linken Position von rechts wieder eingefügt.

Hinweis: Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung. dass sie unterschiedliche Ergebnisse für erg_byte und erg_word erhalten, abhängig vom Datentyp der Eingangsvariablen (BYTE oder WORD), obwohl die Werte der Eingangsvariablen in_byte und in_word gleich sind.

Beispiel in ST:

```
PROGRAM rol_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
erg_byte:=ROL(in_byte,n); (* Ergebnis ist 16#15 *)
erg_word:=ROL(in_word,n); (* Ergebnis ist 16#0114 *)
```

Beispiel in FUP:

Beispiel in AWL:

```
LD      in_byte
ROL     n
ST      erg_byte
```

ROR

IEC Operator: Bitweise Rotation eines Operanden nach rechts.

```
erg = ROR (in, n)
```

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD.

in wird *n* mal um 1 Bit nach rechts verschoben und gleichzeitig das Bit mit der äußersten rechten Position von links wieder eingefügt.

Hinweis: Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung. dass sie unterschiedliche Ergebnisse für erg_byte und erg_word erhalten, abhängig vom Datentyp der Eingangsvariablen (BYTE oder WORD), obwohl die Werte der Eingangsvariablen in_byte und in_word gleich sind.

Beispiel in ST:

```
PROGRAM ror_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
erg_byte:=ROR(in_byte,n); (* Ergebnis ist 16#51 *)
erg_word:=ROR(in_word,n); (* Ergebnis ist 16#4011 *)
```

Beispiel in FUP:

Beispiel in AWL:

```
LD      in_byte
ROR     n
ST      erg_byte
```

16.3.5 Auswahloperatoren

Sämtliche Auswahloperationen lassen sich auch auf Variablen durchführen. Aus Gründen der Anschaulichkeit beschränken sich die hier gezeigten Beispiele auf Konstanten als Operatoren: SEL, MAX, MIN, LIMIT, MUX

SEL

IEC Auswahloperator: Binäre Auswahl. *G* bestimmt, ob *IN0* oder *IN1* der Variablen *OUT* zugewiesen wird.

OUT := SEL(G, IN0, IN1) bedeutet:

OUT := IN0; if G=FALSE

OUT := IN1; if G=TRUE.

Erlaubte Datentypen:

IN0, IN1, OUT: any type

G: BOOL.

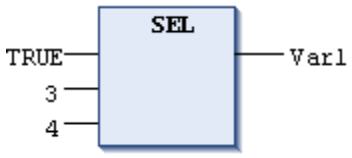
Beispiel in AWL:

```
LD  TRUE
SEL 3,4   (* IN0 = 3, IN1 =4 *)
ST  Var1  (* Ergebnis ist 4 *)
LD  FALSE
SEL 3,4
ST  Var1  (* Ergebnis ist 3 *)
```

Beispiel in ST:

```
Var1:=SEL(TRUE,3,4); (* Ergebnis ist 4 *)
```

Beispiel in FUP:



Hinweis: Ein Ausdruck der IN1 oder IN2 vorgeschaltet ist, wird nicht berechnet, wenn IN0 TRUE ist !

MAX

IEC Auswahloperator: Maximumsfunktion. Liefert von zwei Werten den größten.

```
OUT := MAX(IN0, IN1)
```

IN0, IN1 und OUT können von beliebigem Typ sein.

Beispiel in AWL: (Ergebnis ist 90)

```
LD   90
MAX  30
MAX  40
MAX  77
ST   Var1
```

Beispiel in ST:

```
Var1:=MAX(30,40); (* Ergebnis ist 40 *)
Var1:=MAX(40,MAX(90,30)); (* Ergebnis ist 90 *)
```

Beispiel in FUP:

MIN

IEC Selection Operator: Minimumsfunktion. Liefert von zwei Werten den kleinsten.

```
OUT := MIN(IN0, IN1)
```

IN0, IN1 und OUT können von beliebigem Typ sein.

Beispiel in AWL: (Ergebnis ist 30)

```
LD    90
MIN   30
MIN   40
MIN   77
ST    Var1
```

Beispiel in ST:

```
Var1:=MIN(90,30); (* Ergebnis ist 30 *);
Var1:=MIN(MIN(90,30),40); (* Ergebnis ist 30 *);
```

Beispiel in FUP:

LIMIT

IEC Auswahloperator: Limitierung

```
OUT := LIMIT(Min, IN, Max) bedeutet:
OUT := MIN (MAX (IN, Min), Max)
```

Max ist die obere, Min die untere Schranke für das Ergebnis. Wenn der Wert IN die obere Grenze Max überschreitet, dann liefert LIMIT Max. Wenn IN Min unterschreitet, dann ist das Ergebnis Min.

IN und OUT können von beliebigem Typ sein.

Beispiel in AWL: (Ergebnis ist 80)

```
LD    90
LIMIT 30
      80
ST    Var1
```

Beispiel in ST:

```
Var1:=LIMIT(30,90,80); (* Ergebnis ist 80 *);
```

MUX

IEC Auswahloperator: Operator: Multiplexer

OUT := MUX(K, IN0,...,INn) bedeutet:
OUT := INK.

IN0, ...,INn und OUT können von beliebigem Typ sein. K muss von den Typen BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT oder UDINT sein. MUX wählt aus einer Menge von Werten den K-ten aus. Der erste Wert entspricht K=0. Ist K größer als die Anzahl der weiteren Eingänge (n) , so wird der letzte Wert weiter gegeben (INn).

Beispiel in AWL: (Ergebnis ist 30)

```
LD      0
MUX     30  ,
        40  ,
        50  ,
        60  ,
        70  ,
        80
ST      Var1
```

Beispiel in ST:

```
Var1:=MUX(0,30,40,50,60,70,80); (* Ergebnis ist 30 *);
```

Hinweis: Zum Zweck der Laufzeitoptimierung wird nur der Ausdruck, der INK vorgeschaltet ist, berechnet ! In der Simulation dagegen werden alle Zweige berechnet.

16.3.6 Vergleichsoperatoren

Die folgenden Bit-Shift Operatoren, beschrieben durch die Norm IEC 61131-3, werden unterstützt:

GT, LT, LE, GE, EQ, NE

Es handelt sich um boolsche Operatoren, die jeweils zwei Eingänge (erster und zweiter Operand) vergleichen.

GT

IEC Vergleichsoperator: Größer als.

Ein boolscher Operator mit dem Ergebnis TRUE, wenn der erste Operand größer als der zweite ist. Die Operanden können von jedem beliebigen Basisdatentyp sein.

Beispiel in AWL: (Ergebnis ist FALSE)

```
LD    20
GT    30
ST    Var1
```

Beispiel in ST:

```
VAR1 := 20 > 30 > 40 > 50 > 60 > 70;
```

Beispiel in FUP:

LT

IEC Vergleichsoperator: Kleiner als.

Ein boolscher Operator mit dem Ergebnis TRUE, wenn der erste Operand kleiner als der zweite ist. Die Operanden können von jedem beliebigen Basisdatentyp sein.

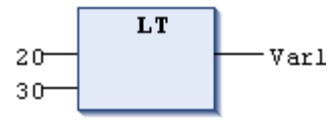
Beispiel in AWL: (Ergebnis ist TRUE)

```
LD    20
LT    30
ST    Var1
```

Beispiel in ST:

```
VAR1 := 20 < 30;
```

Beispiel in FUP:



LE

IEC Vergleichsoperator: Kleiner oder gleich.

Ein Boolscher Operator mit Ergebnis TRUE, wenn der erste Operand kleiner als der zweite Operand oder gleich groß wie der zweite Operand ist. Die Operanden können von jedem beliebigen Basisdatentyp sein.

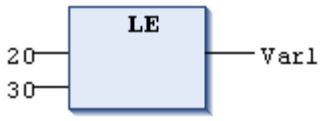
Beispiel in AWL: (Ergebnis ist TRUE)

```
LD    20
LE    30
ST    Var1
```

Beispiel in ST:

```
VAR1 := 20 <= 30;
```

Beispiel in FUP:



GE

IEC Vergleichsoperator: Größer oder gleich

Ein Boolscher Operator mit Ergebnis TRUE, wenn der erste Operand größer als der zweite Operand oder gleich groß wie der zweite Operand ist. Die Operanden können von jedem beliebigen Basisdatentyp sein.

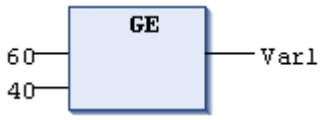
Beispiel in AWL: (Ergebnis ist TRUE)

```
LD    60
GE    40
ST    Var1
```

Beispiel in ST:

```
VAR1 := 60 >= 40;
```

Beispiel in FUP:



EQ

IEC Vergleichsoperator: Gleichheit

Ein Boolscher Operator mit Ergebnis TRUE, wenn die Operanden gleich sind. Die Operanden können von jedem beliebigen Basisdatentyp sein.

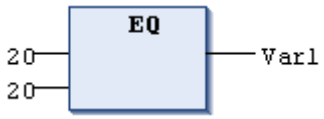
Beispiel in AWL: (Ergebnis ist TRUE)

```
LD    40
EQ    40
ST    Var1
```

Beispiel in ST:

```
VAR1 := 40 = 40;
```

Beispiel in FUP:



NE

IEC Vergleichsoperator: Ungleichheit

Ein Boolscher Operator mit Ergebnis TRUE, wenn die Operanden ungleich sind. Die Operanden können von jedem beliebigen Basisdatentyp sein.

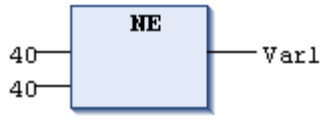
Beispiel in AWL: (Ergebnis ist FALSE)

```
LD    40
NE    40
ST    Var1
```

Beispiel in ST:

```
VAR1 := 40 <> 40;
```

Beispiel in FUP:



16.3.7 Adressoperatoren

ADR und BITADR und der Inhaltsoperator "^" sind in Erweiterung der Norm IEC61131-3 in »PLC Designer« als Adressoperatoren verfügbar.

ADR

Dieser Adressoperator wird in der Norm IEC61131-3 nicht beschrieben.

ADR liefert die Adresse seines Arguments in einem DWORD. Diese Adresse kann an Herstellerfunktionen geschickt und dort wie ein Pointer behandelt werden oder innerhalb des Projektes an einen Pointer zugewiesen werden.

Hinweis: Im Unterschied zu »PLC Designer« V2.3 kann der ADR-Operator nun mit Funktions-, Programm-, Funktionsblock- und Methodennamen verwendet werden und ersetzt damit den INDEXOF Operator. Sehen Sie in diesem Zusammenhang auch: 'Funktionszeiger'. Beachten Sie immer, dass Funktionszeiger zwar an externe Bibliotheken weitergegeben werden können, es jedoch keine Möglichkeit gibt, einen Funktionszeiger innerhalb von »PLC Designer« aufzurufen! Um einen Systemaufruf (Laufzeitsystem) zu ermöglichen, muss die entsprechende Objekteigenschaft (Kategorie "Übersetzen") für das Funktionsobjekt gesetzt sein.

Beispiel in ST:

```
dwVar:=ADR(bVAR);
```

Beispiel in AWL:

```
LD      bVar
ADR
ST      dwVar
```

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

BITADR

Dieser Adressoperator wird in der Norm IEC61131-3 nicht beschrieben.

BITADR liefert den Bitoffset innerhalb des Segments in einem DWORD. Beachten Sie, dass der Offset davon abhängt, ob die Option Byteadressierung in den Zielsystemeinstellungen aktiviert ist oder nicht.

```
VAR
 var1 AT %IX2.3:BOOL;
 bitoffset: DWORD;
END_VAR
```

Beispiel in ST:

```
bitoffset:=BITADR(var1); (* Result if byte addressing=TRUE: 19, if byte addressing=FALSE: 35 *)
```

Beispiel in AWL:

```
LD      Var1
BITADR
ST      bitoffset
```

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

Inhaltsoperator

Dieser Adressoperator wird in der Norm IEC61131-3 nicht beschrieben. Er ermöglicht das Dereferenzieren eines Pointers. Er wird als "^" an den Pointer-Bezeichner angehängt.

Beispiel in ST:

```
pt:POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2:=pt^;
```

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

16.3.8 Aufrufoperatoren

CAL

IEC Operator zum Aufruf eines Funktionsblocks

Mit CAL ruft man in **AWL** die Instanz eines Funktionsblock auf. Nach dem Namen der Instanz eines Funktionsblocks folgt, in runde Klammern gesetzt, die Belegung der Eingabevariablen des Funktionsblocks.

Beispiel:

Aufruf der Instanz *Inst* eines Funktionsblocks mit Belegung der Eingabevariablen *Par1*, *Par2* auf 0 bzw. TRUE.

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

16.3.9 Typkonvertierungs-Operatoren

Es ist nicht erlaubt, von einem "größeren" Typ auf einen "kleineren" implizit zu konvertieren (beispielsweise von INT nach BYTE oder von DINT nach WORD). Wenn man das tun will, muss man spezielle Typkonvertierungen anwenden. Grundsätzlich kann von jedem elementaren Typ zu jeden anderen elementaren Typ konvertiert werden.

Syntax:

<elem.Typ1>_TO_<elem.Typ2>

Beachten Sie bei ...TO_STRING Konvertierungen, dass der String (Zeichenfolge) "linksbündig" generiert wird. Wenn er zu kurz definiert ist, wird von rechts her abgeschnitten.

Folgende Typkonvertierungen werden unterstützt:

BOOL_TO Konvertierungen

TO_BOOL Konvertierungen

Konvertierung ganzzahliger Typen

REAL_TO-/ LREAL_TO Konvertierungen

TIME_TO/TIME_OF_DAY Konvertierungen

DATE_TO/DT_TO Konvertierungen

STRING_TO Konvertierungen

neu gegenüber V2.3:

TRUNC (Konvertierung nach DINT)

TRUNC_INT

ANY_NUM_TO_<numerischer Datentyp>

ANY_TO_<beliebiger Datentyp>

BOOL_TO Konvertierungen

IEC Operator: Konvertierung von Datentyp BOOL zu einem anderen Typ.

Syntax:

BOOL_TO_<data type>

Bei Zahlentypen ist das Ergebnis 1, wenn der Operand TRUE ist, und 0, wenn der Operand FALSE ist.

Beim Typ STRING ist das Ergebnis 'TRUE' bzw. 'FALSE'.

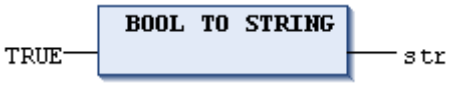
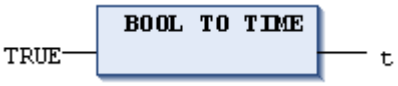
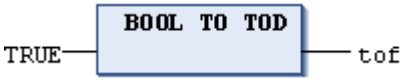
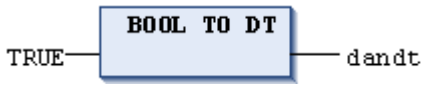
Beispiele in AWL:

LD TRUE BOOL_TO_INT ST i	(*Ergebnis ist 1 *)
LD TRUE BOOL_TO_STRI... ST str	(*Ergebnis ist 'TRUE' *)
LD TRUE BOOL_TO_TIME ST t	(*Ergebnis ist T#1ms *)
LD TRUE BOOL_TO_TOD ST tof	(*Ergebnis ist TOD#00:00:00.001 *)
LD FALSE BOOL_TO_DATE ST dandt	(*Ergebnis ist D#1970-01-01 *)
LD TRUE BOOL_TO_DT ST dandt	(*Ergebnis ist DT#1970-01-01-00:00:01 *)

Beispiele in ST:

i:=BOOL_TO_INT(TRUE);	(* Ergebnis ist 1 *)
str:=BOOL_TO_STRING(TRUE);	(* Ergebnis ist "TRUE" *)
t:=BOOL_TO_TIME(TRUE);	(* Ergebnis ist T#1ms *)
tof:=BOOL_TO_TOD(TRUE);	(* Ergebnis ist TOD#00:00:00.001 *)
dat:=BOOL_TO_DATE(FALSE);	(* Ergebnis ist D#1970 *)
dandt:=BOOL_TO_DT(TRUE);	(* Ergebnis ist DT#1970-01-01-00:00:01 *)

Beispiele in FUP:

	(*Ergebnis ist 1 *)
	(*Ergebnis ist 'TRUE' *)
	(*Ergebnis ist T#1ms *)
	(*Ergebnis ist TOD#00:00:00.001 *)
	(*Ergebnis ist D#1970-01-01 *)
	(*Ergebnis ist DT#1970-01-01-00:00:01 *)

TO_BOOL Konvertierungen

IEC Operator: Konvertierung von another variable type to BOOL

Syntax:

<data type>_TO_BOOL

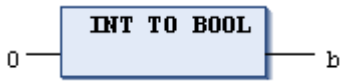
Das Ergebnis ist TRUE, wenn der Operand ungleich 0 ist. Das Ergebnis ist FALSE, wenn der Operand gleich 0 ist.

Beim Typ STRING ist das Ergebnis TRUE, wenn der Operand 'TRUE' ist, ansonsten ist das Ergebnis FALSE.

Beispiele in AWL:

LD 213 BYTE_TO_BOOL ST b	(*Ergebnis ist TRUE *)
LD 0 INT_TO_BOOL ST b	(*Ergebnis ist FALSE *)
LD T#5ms TIME_TO_BOOL ST b	(*Ergebnis ist TRUE *)
LD 'TRUE' STRING_TO_BOOL ST b	(*Ergebnis ist TRUE *)

Beispiele in FUP:

	(*Ergebnis ist TRUE *)
	(*Ergebnis ist FALSE *)
	(*Ergebnis ist TRUE *)
	(*Ergebnis ist TRUE *)

Beispiele in St:

b := BYTE_TO_BOOL(2#11010101);	(* Ergebnis ist TRUE *)
b := INT_TO_BOOL(0);	(* Ergebnis ist FALSE *)
b := TIME_TO_BOOL(T#5ms);	(* Ergebnis ist TRUE *)
b := STRING_TO_BOOL('TRUE');	(* Ergebnis ist TRUE *)

Konvertierung ganzzahliger Typen

Conversion von einem ganzzahligen Zahlentyp zu einem anderen Zahlentyp:

Syntax:

<INT data type>_TO_<INT data type>

Hinweis: Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen. Wenn die zu konvertierende Zahl die Bereichsgrenze überschreitet, dann werden die ersten Bytes der Zahl nicht berücksichtigt.

Beispiel in ST:

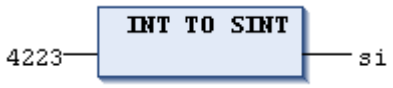
```
si := INT_TO_SINT(4223); (* Ergebnis ist 127 *)
```

Wenn sie die Integer-Zahl 4223 (16#107f in Hexadezimaldarstellung) in eine SINT-Variable speichern, dann enthält diese die Zahl 127 (16#7f in Hexadezimaldarstellung).

Beispiel in AWL:

```
LD          4223
INT_TO_SINT
ST          si
```

Beispiel in FUP:



REAL_TO-/ LREAL_TO Konvertierungen

IEC Operator: Konvertierung vom Typ REAL bzw. LREAL zu einem anderen Typ.

Es wird nach oben oder unten auf einen ganzzahligen Wert gerundet und in den entsprechenden Typen gewandelt. Ausgenommen davon sind die Typen STRING, BOOL, REAL und LREAL.

Hinweis: Wenn ein REAL oder LREAL in SINT, USINT, INT, UINT, DINT, UDINT, LINT oder ULINT konvertiert wird und der Wert der REAL/LREAL-Zahl außerhalb des Wertebereichs des Integers liegt, erhält man ein undefiniertes Ergebnis das vom Zielsystem abhängt.

Auch ein Ausnahmefehler ist dann möglich! Um zielsystemunabhängigen Code zu erhalten, müssen Wertebereichüberschreitungen über die Applikation abgefangen werden. Wenn die REAL/LREAL Zahl innerhalb des Bereichs liegt, arbeitet die Konvertierung auf allen Systemen gleich.

Beachten Sie bei der Konvertierung in den Typ STRING, dass die Gesamtkommastellenzahl auf 16 begrenzt ist. Enthält die (L)REAL-Zahl mehr Stellen, wird die sechzehnte Stelle gerundet und so im string dargestellt. Wenn der STRING für die Zahl zu kurz definiert ist, wird von rechts her entsprechend abgeschnitten.

Hinweis: Bei der Typkonvertierung von größere auf kleinere Typen können Informationen verloren gehen.

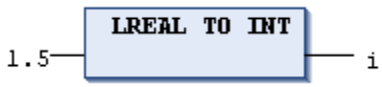
Beispiel in ST:

```
i := REAL_TO_INT(1.5); (* Ergebnis ist 2 *)
j := REAL_TO_INT(1.4); (* Ergebnis ist 1 *)
i := REAL_TO_INT(-1.5); (* Ergebnis ist -2 *)
j := REAL_TO_INT(-1.4); (* Ergebnis ist -1 *)
```

Beispiel in AWL:

```
LD          2.7
REAL_TO_INT
ST          i
```

Beispiel in FUP:



TIME_TO/TIME_OF_DAY Konvertierungen

IEC Operator: Konvertierung vom Typ TIME bzw. TIME_OF_DAY zu einem anderen Typ.

Syntax:

<TIME-Datentyp>_TO_<Datentyp>

Intern wird die Zeit in einem DWORD in Millisekunden abgespeichert (bei TIME_OF_DAY seit 00:00 Uhr). Dieser Wert wird konvertiert.

Beim Typ STRING ist das Ergebnis die Zeitkonstante.

Hinweis: Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen.

Beispiele in AWL:

`LD T#12ms`
`TIME_TO_STTI...`
`ST str`
	(* Ergebnis ist 'T#12ms' *)

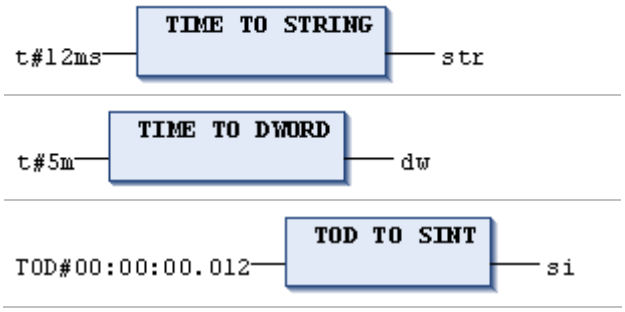
`LD T#300000ms`
`TIME_TO_DWORD`
`ST dw`
	(* Ergebnis ist 300000 *)

`LD TOD#00:00:00.012`
`TIME TO SINT`
`ST si`
	(* Ergebnis ist 12 *)

Beispiele in ST:

str :=TIME_TO_STRING(T#12ms);	(* Ergebnis ist T#12ms *)
dw:=TIME_TO_DWORD(T#5m);	(* Ergebnis ist 300000 *)
si:=TOD_TO_SINT(TOD#00:00:00.012);	(* Ergebnis ist 12 *)

Beispiele in FUP:



DATE_TO/DT_TO Konvertierungen

IEC Operator: Konvertierung vom Typ DATE bzw. DATE_AND_TIME zu einem anderen Typ:

Syntax:

<DATE-Datentyp>_TO_<Datentyp>

Intern wird das Datum in einem DWORD in Sekunden seit dem 1.Januar 1970 abgespeichert. Dieser Wert wird konvertiert.

Beim Typ STRING ist das Ergebnis die Datumskonstante.

Hinweis: Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen.

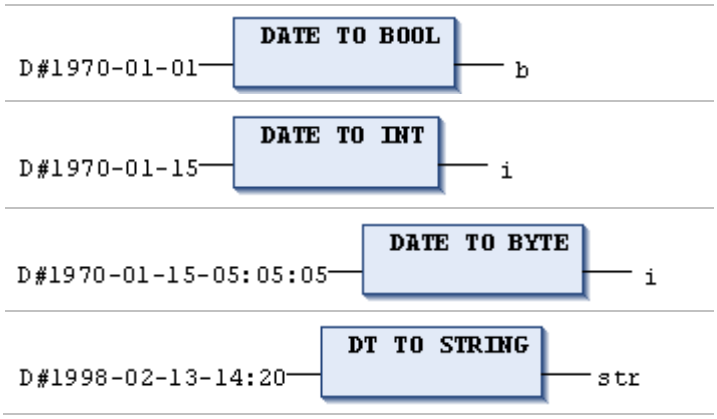
Beispiele in AWL:

LD D#1970-01-01 DATE_TO_BOOL ST b	(* Ergebnis ist FALSE *)
LD D#1970-01-01 DATE_TO_INT ST i	(* Ergebnis ist 29952 *)
LD D#1970-01-15-05:05: DATE_TO_BYTE ST byt	(* Ergebnis ist 129 *)
LD D#1998-02-13-14:20 DATE_TO_STRI... ST str	(* Ergebnis ist 'DT#1998-02-13-14:20' *)

Beispiele in ST:

b :=DATE_TO_BOOL(D#1970-01-01);	(* Ergebnis ist FALSE *)
i :=DATE_TO_INT(D#1970-01-15);	(* Ergebnis ist 29952 *)
byt :=DT_TO_BYTE(DT#1970-01-15-05:05:05);	(* Ergebnis ist 129 *)
str:=DT_TO_STRING(DT#1998-02-13-14:20);	(* Ergebnis ist 'DT#1998-02-13-14:20' *)

Beispiele in FUP:



STRING_TO Konvertierungen

IEC Operator: Konvertierung vom Typ STRING zu einem anderen Typ.

Die Konvertierung arbeitet nach dem Standard C-Kompilierungsmechanismus: STRING nach INT und dann INT nach BYTE. Das höhere Byte wird abgeschnitten. d.h. es ergeben sich nur Werte von 0 - 255.

Syntax:

STRING_TO_<Datentyp>

Der Operand vom Typ STRING muss einen gültigen Wert des Zieltyps haben, sonst ist das Ergebnis 0.

Hinweis: Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen.

Beispiele in AWL:

`LD 'TRUE'`
`STRING_TO_BOOL`
`ST b`
	(* Ergebnis ist TRUE *)

`LD 'abc34'`
`STRING_TO_WORD`
`ST w`
	(* Ergebnis ist 0 *)

`LD 't#117ms'`
`STRING_TO_TIME`
`ST t`
	(* Ergebnis ist T#117ms *)

`LD '500'`
`STRING_TO_BYTE`
`ST bv`
	(* Ergebnis ist 244 *)

Beispiele in ST:

b :=STRING_TO_BOOL('TRUE');	(* Ergebnis ist TRUE *)
w :=STRING_TO_WORD('abc34');	(* Ergebnis ist 0 *)
t :=STRING_TO_TIME('T#127ms');	(* Ergebnis ist T#127ms *)
bv :=STRING:TO_BYTE('500');	(* Ergebnis ist 244 *)

Beispiele in FUP:

	(* Ergebnis ist TRUE *)
	(* Ergebnis ist 0 *)
	(* Ergebnis ist T#127ms *)
	(* Ergebnis ist 244 *)

TRUNC

IEC Operator: Konvertierung vom Typ REAL zum Typ DINT. Es wird nur der Betrag des ganzzahligen Anteils der Zahl genommen.

Hinweis: In »PLC Designer« V2.3 konvertiert der TRUNC Operator von REAL nach INT. Wenn ein V2.3 Projekt importiert wird, wird TRUNC deshalb automatisch durch TRUNC_INT ersetzt.

Beispiel in AWL:

```
LD      1.9
TRUNC
ST      diVar
```

Beispiele in ST:

```
diVar:=TRUNC(1.9); (* Ergebnis ist 1 *)
diVar:=TRUNC(-1.4); (* Ergebnis ist -1 *).
```

TRUNC_INT

IEC Operator: Konvertierung vom Typ REAL zum Typ INT. Es wird nur der Betrag des ganzzahligen Anteils der Zahl genommen.

Hinweis: TRUNC_INT entspricht dem Operator TRUNC in »PLC Designer« V2.3 und wird beim Import von V2.3-Projekten automatisch an dessen Stelle verwendet. Beachten Sie die veränderte Funktion von TRUNC in V3.

Beispiel in AWL:

```
LD      1.9
TRUNC_INT
ST      iVar
```

Beispiele in ST:

```
iVar:=TRUNC_INT(1.9); (* Ergebnis ist 1 *)
iVar:=TRUNC_INT(-1.4); (* Ergebnis ist -1 *).
```

ANY_TO Konvertierungen

Konvertierung eines beliebigen Datentyps ANY, bzw. spezifischer eines numerischen Datentyps ANY_NUM, zu einem anderen Typ. Eine sinnvolle Auswahl der Datentypen wird vorausgesetzt.

Syntax:

ANY_NUM_TO_<numerischer Datentyp>

ANY_TO_<beliebiger Datentyp>

Beispiel:

Konvertierung von Datentyp REAL nach INT:

```
re : REAL := 1.234;
i : INT := ANY_TO_INT(re)
```

16.3.10 Numerische Operatoren

Die folgenden numerischen Operatoren, beschrieben durch die Norm IEC 61131-3, werden unterstützt::

ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN, EXPT

ABS

Numerischer IEC Operator: Liefert den Absolutwert einer Zahl. ABS(-2).

Ein- und Ausgangsvariablen können von einem beliebigen numerischen Basisdatentyp sein.

Beispiel in AWL: (Result in i is 2)

```
LD      -2
ABS
ST      i
```

Beispiel in ST:

```
i:=ABS(-2);
```

Beispiel in FUP:

SQRT

Numerischer IEC Operator: Liefert die Quadratwurzel einer Zahl.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 4)

```
LD    16
SQRT
ST    q
```

Beispiel in ST:

```
q:=SQRT(16);
```

Beispiel in FUP:

LN

Numerischer IEC Operator: Liefert den natürlichen Logarithmus einer Zahl

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 3.80666)

```
LD    45
LN
ST    q
```

Beispiel in ST:

```
q:=LN(45);
```

Beispiel in FUP:

LOG

Numerischer IEC Operator: Liefert den Logarithmus zur Basis 10 einer Zahl.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 2.49762 *)

```
LD      314.5
LOG
ST      q
```

Beispiel in ST:

```
q:=LOG(314.5);
```

Beispiel in FUP:

EXP

Numerischer IEC Operator: Liefert die Exponentialfunktion.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 7.389056099)

```
LD      2
EXP
ST      q
```

Beispiel in ST:

```
q:=EXP(2);
```

Beispiel in FUP:

SIN

Numerischer IEC Operator: RLiefert den Sinus einer Zahl. Der Wert wird in Bogenmaß errechnet.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

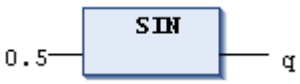
Beispiel in AWL: (Ergebnis in q ist 0.479426 *)

```
LD      0.5
SIN
ST      q
```

Beispiel in ST:

```
q:=SIN(0.5);
```

Beispiel in FUP:



COS

Numerischer IEC Operator: Liefert den Cosinus einer Zahl. Der Wert wird in Bogenmaß errechnet.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (* Ergebnis in q ist 0.877583 *)

```
LD      0.5
COS
ST      q
```

Beispiel in ST:

```
q:=COS(0.5);
```

Beispiel in FUP:

TAN

Numerischer IEC Operator: Liefert den Tangens einer Zahl. Der Wert wird in Bogenmaß errechnet.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 0.546302 *)

```
LD      0.5
TAN
ST      q
```

Beispiel in ST:

```
q:=TAN(0.5);
```

Beispiel in FUP:

ASIN

Numerischer IEC Operator: Liefert den Arcus Sinus (Umkehrfunktion von Sinus) einer Zahl. Der Wert wird in Bogenmaß errechnet.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 0.523599 *)

```
LD    0.5
ASIN
ST    q
```

Beispiel in ST:

```
q:=ASIN(0.5);
```

Beispiel in FUP:

ACOS

Numerischer IEC Operator: Liefert den Arcus Cosinus (Umkehrfunktion von Cosinus) einer Zahl. Der Wert wird in Bogenmaß errechnet.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 1.0472)

```
LD    0.5
ACOS
ST    q
```

Beispiel in ST:

```
q:=ACOS(0.5);
```

Beispiel in FUP:

ATAN

Numerischer IEC Operator: Liefert den Arcus Tangens (Umkehrfunktion von Tangens) einer Zahl. Der Wert wird in Bogenmaß errechnet.

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis in q ist 0.463648)

```
LD      0.5
ATAN
ST      q
```

Beispiel in ST:

```
q:=ATAN(0.5);
```

Beispiel in FUP:

EXPT

Numerischer IEC Operator: Potenzierung einer Variablen mit einer anderen.

```
OUT = IN1IN2.
```

Die Eingangsvariable kann von einem beliebigen numerischen Basisdatentyp sein, die Ausgangsvariable muss vom Typ REAL oder LREAL sein.

Beispiel in AWL: (Ergebnis ist 49)

```
LD      7
EXPT    2
ST      Var1
```

Beispiel in ST:

```
var1 := EXPT(7,2);
```

Beispiel in FUP:

16.3.11 IEC-erweiternde Funktionen

Neben den in der IEC-Norm beschriebenen Operatoren, unterstützt »PLC Designer« folgende Funktionen:

ADR

BITADR

INDEXOF

SIZEOF

ANDN

ORN

XORN

__DELETE

__ISVALIDREF

__NEW

__QUERYINTERFACE

__QUERYPOINTER

__DELETE

Dieser Operator wird in der Norm IEC61131-3 nicht beschrieben.

Hinweis: Aus Kompatibilitätsgründen muss die Compiler-Version >= 3.3.2.0 sein.

Der Operator __DELETE gibt den Speicher von Instanzen, die mit _NEW dynamisch erzeugt wurden, wieder frei. __DELETE hat keinen Rückgabewert und der Operand wird nach dieser Operation auf 0 gesetzt.

Mit __NEW wird Speicher alloziert.

Syntax:

__DELETE (<Pointer>)

Falls <Pointer> auf einen Funktionsblock zeigt, wird die zugehörige Methode FB_EXIT aufgefufen, bevor der Pointer auf 0 gesetzt wird.

Beispiel mit Funktionsblock:

```
FUNCTION_BLOCK FBDynamic
VAR_INPUT
in1, in2 : INT;
END_VAR
VAR_OUTPUT
out : INT;
END_VAR
VAR
test1 : INT := 1234;
_inc : INT := 0;
_dut : POINTER TO DUT;
neu : BOOL;
END_VAR
out := in1 + in2;

METHOD FB_Exit : BOOL
VAR_INPUT
bInCopyCode : BOOL;
END_VAR
__Delete(_dut);

METHOD FB_Init : BOOL
VAR_INPUT
bInitRetains : BOOL;
bInCopyCode : BOOL;
END_VAR
_dut := __NEW(DUT);

METHOD INC : INT
VAR_INPUT
END_VAR
_inc := _inc + 1;
INC := _inc;

PLC_PRG(PRG)
VAR
pFB : POINTER TO FBDynamic;
```

```
bInit: BOOL := TRUE;
bDelete: BOOL;
loc : INT;
END_VAR
IF (bInit) THEN
pFB := __NEW(FBDynamic);
bInit := FALSE;
END_IF
IF (pFB <> 0) THEN
pFB^(in1 := 1, in2 := loc, out => loc);
pFB^.INC();
END_IF
IF (bDelete) THEN
__DELETE(pFB);
END_IF
```

__ISVALIDREF

Dieser Operator wird in der Norm IEC61131-3 nicht beschrieben. Mit seiner Hilfe kann geprüft werden, ob eine Referenz auf einen gültigen Wert verweist. Wie er anzuwenden ist und ein Beispiel finden Sie in der Beschreibung des Datentyps Referenz.

__NEW

Dieser Operator wird in der Norm IEC61131-3 nicht beschrieben.

Hinweis: Aus Kompatibilitätsgründen muss die Compiler-Version >= 3.3.2.0 sein.

__NEW alloziert Speicher für Instanzen von Funktionsblöcken oder Arrays von Standard-Datentypen. Der Operator gibt einen Pointer auf das passend typisierte Objekt zurück, der nicht 0 ist. Falls der Operator nicht in einer Zuweisung verwendet wird, wird eine Fehlermeldung ausgegeben.

Scheitert der Versuch Speicher zu allozieren, wird von __NEW 0 zurückgegeben.

Mit __DELETE wird der Speicher wieder freigegeben.

Syntax:

__NEW (<Typ>, [<Größe>])

Der Operator erzeugt eine Instanz auf den spezifizierten Typen <Typ> und gibt einen Pointer auf diese Instanz zurück. Anschließend wird die Initialisierung der Instanz aufgerufen.

Fall es sich bei <Typ> um einen skalaren Typen handelt, wird zusätzlich der optionale Operand <Größe> gesetzt. Der Operator erzeugt dann einen Array des Typs <Type>und der Größe <Größe>.

Beispiel:

```
pScalarType := __New(ScalarType, length);
```

Hinweis: Für dynamisch erzeugte Instanzen ist kein Copy-Code-Verfahren im Online-Modus möglich!

Aus diesem Grund können Funktionsblöcke aus Bibliotheken, die nicht geändert werden können, und Funktionsblöcke mit dem Attribut 'enable_dynamic_creation' auf den __NEW-Operator angewendet werden. Falls ein Funktionsblock mit dieser Kennzeichnung geändert werden soll, so dass Copy-Code notwendig wird, wird eine Fehlermeldung ausgegeben.

Hinweis: Der Code für Speicheralloktion muss wiedereintrittsfähig sein.

Ein Semaphor (SysSemEnter) kann benutzt werden, um zu vermeiden dass zwei Tasks gleichzeitig versuchen Speicher zu allozieren. Das hat zur Folge, dass die umfangreiche Nutzung von __NEW einen höheren Jitter verursacht.

Example mit skalarem Typ:

```
TYPE DUT :
STRUCT
a,b,c,d,e,f : INT;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
pDut : POINTER TO DUT;
bInit: BOOL := TRUE;
bDelete: BOOL;
END_VAR
```

```
IF (bInit) THEN
pDut := __NEW(DUT);
bInit := FALSE;
END_IF
IF (bDelete) THEN
__DELETE(pDut);
END_IF
```

Beispies mit Funktionsblock:

```
FBDynamic(FP)
{attribute 'enable_dynamic_creation'}
FUNCTION_BLOCK FBDynamic
VAR_INPUT
in1, in2 : INT;
END_VAR
VAR_OUTPUT
out : INT;
END_VAR
VAR
test1 : INT := 1234;
_inc : INT := 0;
_dut : POINTER TO DUT;
neu : BOOL;
END_VAR
out := in1 + in2;

PLC_PRG(PRG)
VAR
pFB : POINTER TO FBDynamic;
loc : INT;
bInit: BOOL := TRUE;
bDelete: BOOL;
END_VAR
IF (pFB <> 0) THEN
pFB^(in1 := 1, in2 := loc, out => loc);
pFB^.INC();
END_IF
IF (bDelete) THEN
__DELETE(pFB);
END_IF
```

Beispiel mit Array:

```
PLC_PRG(PRG)
VAR
bInit: BOOL := TRUE;
bDelete: BOOL;
pArrayBytes : POINTER TO BYTE;
pArrayDuts : POINTER TO BYTE;
test: INT;
parr : POINTER TO BYTE;
END_VAR
IF (bInit) THEN
pArrayBytes := __NEW(BYTE, 25);
bInit := FALSE;
END_IF
IF (pArrayBytes <> 0) THEN
pArrayBytes[24] := 125;
test := pArrayBytes[24];
END_IF
IF (bDelete) THEN
__DELETE(pArrayBytes);
END_IF
```

__QUERYINTERFACE

Dieser Operator wird in der Norm IEC61131-3 nicht beschrieben.

__QUERYINTERFACE ermöglicht zur Laufzeit eine Typkonvertierung einer Interface-Referenz ener anderen. Der Operator liefert ein Ergebnis vom Typ BOOL zurück. TRUE bedeutet, dass die Konvertierung erfolgreich durchgeführt wurde.

Hinweis: Aus Kompatibilitätsgründen muss die Definition der zu konvertierenden Referenz eine Erweiterung des Basis-Interface __SYSTEM.IQueryInterface sein und die Compiler-Version muss >= 3.3.0.10 sein.

Syntax:

__QUERYINTERFACE(<ITF_Source>, < ITF_Dest>)

Dieser Operator bekommt als 1. Operanden eine Interface-Referenz oder eine FB-Instanz, als 2. Operanden eine Interface-Referenz mit dem gewünschten Zieltypen. Nach Abarbeiten von __QUERYINTERFACE enthält ITF_Dest eine Referenz auf das gewünschte Interface, wenn das von ITF_Source referenzierte Objekt das Interface implementiert. Dann war die Konvertierung erfolgreich und das Ergebnis des Operators liefert TRUE. In allen anderen Fällen wird FALSE geliefert.

Voraussetzung für die explizite Konvertierung ist, dass sowohl das ITF_Source als auch ITF_Dest eine Ableitung vom Interface __System.IQueryInterface sind. Dieses Interface steht implizit zur Verfügung und benötigt keine Bibliothek.

Beispiel:

```
INTERFACE ItfBase EXTENDS __System.IQueryInterface
METHOD mbase : BOOL
END_METHOD

INTERFACE ItfDerived1 EXTENDS ItfBase
METHOD mderived1 : BOOL
END_METHOD

INTERFACE ItfDerived2 EXTENDS ItfBase
METHOD mderived2 : BOOL
END_METHOD

PROGRAMM POU
VAR
itfderived1 : ItfDerived1;
itfderived2 : ItfDerived2;
bTest1, bTest2, xResult1, xResult2: BOOL;
END_VAR

xResult1 := __QUERYINTERFACE(itfbase, itfderived1); // variablen vom Type ItfBase bzw ItfDerived1
xResult2 := __QUERYINTERFACE(itfbase, itfderived2); // variablen vom Type ItfBase bzw ItfDerived2
IF (xResult1 = TRUE) THEN
bTest1 := itfderived1.mderived1();
ELSIF xResult2 THEN
bTest2 := itfderived2.mderived2();
END_IF
```

__QUERYPOINTER

Dieser Operator wird in der Norm IEC61131-3 nicht beschrieben.

__QUERYPOINTER ermöglicht zur Laufzeit die Typkonvertierung einer Interface-Reference eines Funktionsbausteins auf einen Pointer. Der Operator liefert ein Ergebnis vom Typ BOOL zurück. TRUE bedeutet, dass die Konvertierung erfolgreich durchgeführt wurde.

Hinweis: Aus Kompatibilitätsgründen muss die Definition des zu konvertierenden Pointers eine Erweiterung des Basis-Interface __SYSTEM.IQueryInterface sein und die Compiler-Version muss >= 3.3.0.10 sein.

Syntax:

__QUERYPOINTER (<ITF_Source>, < Pointer_Dest>)

Dieser Operator bekommt als 1. Operanden eine Interface-Referenz oder eine FB-Instanz mit den gewünschten Zieltypen und als 2. Operanden einen Pointer. Nach Abarbeiten von __QUERYPOINTER enthält Pointer_Dest den Pointer auf diejenige Referenz oder Instanz eines Funktionsbausteins, auf die die Interface-Referenz ITF_Source aktuell verweist. Pointer_Dest ist nicht getypt und kann auf einen beliebigen Typ gecastet werden. Der Typ muss vom Programmierer selbst sichergestellt werden. Beispielsweise könnte das Interface eine Methode anbieten, die einen Typ-Code zurück liefert.

Voraussetzung für die explizite Konvertierung ist, dass das ITF_Source eine Ableitung vom Interface __System.IQueryInterface ist. Dieses Interface steht implizit zur Verfügung und benötigt keine Bibliothek.

Beispiel:

```
INTERFACE ItfBase EXTENDS __System.IQueryInterface
METHOD mbase : BOOL
END_METHOD

INTERFACE ItfDerived1 EXTENDS ItfBase
METHOD mderived : BOOL
END_METHOD

FUNCTION_BLOCK FBVariante IMPLEMENTS ITFDerived

PROGRAMM POU
VAR
itfderived : ItfDerived;
insV : FBVariante;
xResult, xTest : BOOL;
pVar: POINTER TO DWORD;

END_VAR

itfderived := insV;
xResult := __QUERYPOINTER(itfderived, pVar);
IF xResult THEN
    xTest := pVar.mderived();
END_IF
```

ADR

Dieser Adressoperator wird in der Norm IEC61131-3 nicht beschrieben.

ADR liefert die Adresse seines Arguments in einem DWORD. Diese Adresse kann an Herstellerfunktionen geschickt und dort wie ein Pointer behandelt werden oder innerhalb des Projektes an einen Pointer zugewiesen werden.

Hinweis: Im Unterschied zu »PLC Designer« V2.3 kann der ADR-Operator nun mit Funktions-, Programm-, Funktionsblock- und Methodennamen verwendet werden und ersetzt damit den INDEXOF Operator. Sehen Sie in diesem Zusammenhang auch: 'Funktionszeiger'. Beachten Sie immer, dass Funktionszeiger zwar an externe Bibliotheken weitergegeben werden können, es jedoch keine Möglichkeit gibt, einen Funktionszeiger innerhalb von »PLC Designer« aufzurufen! Um einen Systemaufruf (Laufzeitsystem) zu ermöglichen, muss die entsprechende Objekteigenschaft (Kategorie "Übersetzen") für das Funktionsobjekt gesetzt sein.

Beispiel in ST:

```
dwVar:=ADR(bVAR);
```

Beispiel in AWL:

```
LD      bVar
ADR
ST      dwVar
```

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

BITADR

Dieser Adressoperator wird in der Norm IEC61131-3 nicht beschrieben.

BITADR liefert den Bitoffset innerhalb des Segments in einem DWORD. Beachten Sie, dass der Offset davon abhängt, ob die Option Byteadressierung in den Zielsystemeinstellungen aktiviert ist oder nicht.

```
VAR
 var1 AT %IX2.3:BOOL;
 bitoffset: DWORD;
END_VAR
```

Beispiel in ST:

```
bitoffset:=BITADR(var1); (* Result if byte addressing=TRUE: 19, if byte addressing=FALSE: 35 *)
```

Beispiel in AWL:

```
LD      Var1
BITADR
ST      bitoffset
```

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

INDEXOF

Dieser arithmetische Operator ist nicht von der Norm IEC61131-3 vorgeschrieben.

Mit dieser Funktion kann der interne Index eines Bausteins ermittelt werden.

Beispiel in ST:

```
var1 := INDEXOF(POU2);
```

SIZEOF

Dieser arithmetische Operator ist nicht von der Norm IEC61131-3 vorgeschrieben.

Er kann verwendet werden, um die Anzahl Bytes zu bestimmen, die von der angegebenen Variablen x benötigt werden

Der SIZEOF Operator liefert immer einen vorzeichenlosen Wert. Der Typ der Rückgabevariablen paßt sich der gefundenen Größe von Variable x an.

Rückgabewert von SIZEOF(x)	Datentyp der Konstanten, die implizit für die gefundene Größe verwendet wird
0 <= Größe von x < 256	USINT
256 <= Größe von x < 65536	UINT
65536 <= Größe von x < 4294967296	UDINT
4294967296 <= Größe von x	ULINT

Beispiel in ST:

```
var1 := SIZEOF(arr1);  (* d.h.:  var1:=USINT#10; *)
```

Beispiel in AWL: (Ergebnis ist 10)

arr1:ARRAY[0..4] OF INT;

Var1:INT;

```
LD      arr1
SIZEOF
ST      Var1
```

ANDN

Noch nicht verfügbar.

Dieser IEC Bitstring Operator wird nicht von der Norm IEC61131-3 beschrieben.

ORN

Noch nicht verfügbar.

Dieser IEC Bitstring Operator wird nicht von der Norm IEC61131-3 beschrieben.

XORN

Noch nicht verfügbar.

Dieser IEC Bitstring Operator wird nicht von der Norm IEC61131-3 beschrieben.

Namensraum-Operatoren

In Erweiterung der Norm IEC 61131-3 Operatoren gibt es einige Möglichkeiten, den Zugriff auf Variablen oder Module eindeutig zu gestalten auch wenn der gleiche Variablen- oder Modulname im Projekt mehrmals verwendet wird. Um den jeweils gültigen Namensraum zu definieren, können die folgenden Namensraum-Operatoren verwendet werden:

"." (Globaler Namensraum-Operator)

"<Name einer Globalen Variablenliste>."

"<Name einer Bibliothek>."

"<Name einer Enumeration>."

Globaler Namensraum-Operator

Namensraum-Operator: Erweiterung der Norm IEC 61131-3.

Ein Instanzpfad, der mit einem Punkt "." beginnt, öffnet immer einen globalen Namensraum. Wenn es also eine lokale Variable mit dem gleichen Namen "<varname>" gibt wie eine globale Variable, wird mit ".<varname>" die globale Variable angesprochen.

Namensraum für Globale Variablenlisten

Namensraum-Operator: Erweiterung der Norm IEC 61131-3.

Der Name einer globalen Variablenliste (GVL) kann als Namensraumbezeichner für die in der Liste definierten Variablen verwendet werden. Somit ist es möglich, Variablen mit gleichen Namen in verschiedenen globalen Variablenlisten zu verwenden und dennoch eindeutig auf eine bestimmte Variable zuzugreifen, indem dem Variablennamen der Name der Liste - getrennt durch einen Punkt "." - vorangestellt wird:

Syntax:

"<global variable list name>.<variable>".

Beispiel:

Die globalen Variablenlisten "globlist1" und "globlist2" enthalten jeweils eine Variable "varx". In der folgenden Code-Zeile wird varx aus Liste globlist2 in varx aus Liste globlist1 kopiert:

```
globlist1.varx := globlist2.varx;
```

Wenn eine Variable, die in mehreren globalen Variablenlisten deklariert ist, ohne den vorangestellten Listennamen referenziert wird, wird eine Fehlermeldung ausgegeben.

Bibliotheks-Namensraum

Namensraum-Operator: Erweiterung der Norm IEC 61131-3.

Der Bibliotheksname kann verwendet werden, um einen Bibliotheksbaustein eindeutig anzusprechen. Beispiel: Wenn eine Bibliothek, die in einem Projekt eingebunden ist, einen Baustein "fun1" enthält und es auch im Projekt lokal bereits einen Baustein "fun1" gibt, kann der Bibliotheksname dem Bausteinnamen vorangestellt werden - getrennt durch einen Punkt "." - um den Zugriff eindeutig zu machen.

Syntax:

<namespace>.<module name>, e.g. "lib1.fun1".

Standardmäßig heißt der "Namensraum" einer Bibliothek genauso wie die Bibliothek, es kann jedoch auch eine andere Bezeichnung dafür vergeben werden. Diese kann bereits beim Erstellen eines Bibliotheksprojekts in den Projektinformationen eingetragen werden, oder später im Eigenschaften-Dialog einer eingebundenen Bibliothek.

Beispiel:

Es gibt eine Funktion "fun1" in Bibliothek "lib" und eine lokal im Projekt deklarierte Funktion "fun1". Standardmäßig heißt der Namensraum der Bibliothek "lib":

```
res1 := fun(in := 12); // Aufruf der Projekt-Funktion fun
res2 := lib.fun(in := 12); // Aufruf der Bibliotheks-Funktion fun
```

Enumerations-Namensraum

Namensraum-Operator: Erweiterung der Norm IEC 61131-3.

Der TYPE-Name einer Enumeration (Aufzählung) kann für einen eindeutigen Zugriff auf eine Enumerations-Konstante verwendet werden. Somit können Konstanten gleichen Namens in mehreren Enumerationen verwendet werden.

Der Enumerationsname wird dem Konstantennamen - getrennt durch einen Punkt "." - vorangestellt.

Syntax:

<enumeration name>.<constant name>"

Beispiel:

Die Konstante "Blue" ist sowohl eine Komponente der Enumeration "Colors" als auch eine der Enumeration "Feelings".

```
color := Colors.Blue;  // Zugriff auf Komponente Blue in Enumeration Colors
feeling := Feelings.Blue; // Zugriff auf Komponente Blue in Enumeration Feelings
```

16.4 Operanden

Folgendes kann in »PLC Designer« als Operand verwendet werden:

- Konstante (BOOL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME, Zahl, REAL/LREAL, STRING, Getypte Konstanten)
- Variable
- Adresse
- Funktion

16.4.1 Konstanten

BOOL-Konstanten

BOOL Konstanten sind die Wahrheitswerte TRUE und FALSE.

Siehe auch: Datentyp BOOL

TIME-Konstanten

TIME Konstanten werden insbesondere benutzt, um die Standard Timer-Module zu bedienen. Neben der Zeitkonstanten TIME, die die Größe von 32 Bit hat und der IEC-Norm 61131-3 entspricht, wird normerweiternd auch LTIME als Zeitbasis für hochauflösende Timer unterstützt. LTIME hat eine Größe von 64 Bit und eine Auflösung in Nanosekunden.

Syntax für eine Zeitkonstante:

t#<time declaration>

Anstelle von "t#" gelten auch folgende Schreibweisen: "T#", "time", "TIME".

Die Zeit-Deklaration kann die folgenden Zeiteinheiten verwenden. Diese müssen in der folgenden Reihenfolgen hintereinander angeordnet werden, es ist jedoch nicht erforderlich, alle Einheiten anzugeben.

"d": Tage (days)

"h": Stunden (hours)

"m": Minuten

"s": Sekunden

"ms": Millisekunden

Beispiele für korrekte Zeitkonstanten in einer ST-Zuweisung:

TIME1 := T#14ms;	
TIME1 := T#100S12ms;	(*Überlauf in der höchsten Komponente ist erlaubt*)
TIME1 := t#12h34m15s;	

Beispiele für unkorrekte Verwendung:

TIME1 := t#5m68s;	(* Überlauf bei einer niedrigeren Stelle *)
TIME1 := 15ms;	(* T# fehlt *)
TIME1 := t#4ms13d;	(* unkorrekte Reihenfolge der Zeitangaben *)

Syntax für LTIME Konstante:

LTIME#<Zeitdeklaration>

Die Zeitdeklaration kann die gleichen Zeiteinheiten verwenden, wie für die TIME-Konstante beschrieben, s.o., und zusätzlich folgende:

"us" : Mikrosekunden

"ns" : Nanosekunden

Beispiele für korrekte Verwendung von LTIME-Konstanten in einer ST-Zuweisung:

LTIME1 := LTIME#1000d15h23m12s34ms2us44ns	
LTIME1 := LTIME#3445343m3424732874823ns	

siehe auch: Zeitdatentypen

DATE-Konstanten

Diese Konstanten können verwendet werden, um Datumsangaben zu machen.

Syntax:

dt#<Datumsdeklaration>

Anstelle von "d#" gilt auch folgende Schreibweise: "D#", "date", "DATE".

Die Datumsdeklaration muss im Format "<Jahr-Monat-Tag>" angegeben werden.

DATE (kurz D) Werte werden intern wie DWORD behandelt. Die Zeit wird in Sekunden angegeben, wobei ab dem 1. Januar 1970 um 00:00 Uhr gerechnet wird.

Beispiele:

```
DATE#1996-05-06
d#1972-03-29
```

siehe auch: Zeitdatentypen

TIME_OF_DAY Konstanten

Mit einer solchen Konstanten können Uhrzeiten angegeben werden.

Syntax:

tod#<Uhrzeitdeklaration>

Anstelle von "tod#" ist auch folgende Schreibweise gültig: "TOD#", "time_of_day", "TIME_OF_DAY".

Die Uhrzeitdeklaration muss im Format "<Stunde:Minute:Sekunde>" angegeben werden.

Sekunden können dabei als reelle Zahlen angegeben werden, es können also auch Sekundenbruchteile angegeben werden.

TIME_OF_DAY (kurz TOD) Werte werden intern wie DWORD behandelt. Die Zeit wird in Millisekunden angegeben, wobei bei ab 00:00 Uhr gerechnet wird.

Beispiele:

```
TIME_OF_DAY#15:36:30.123
tod#00:00:00
```

siehe auch: Zeitdatentypen

DATE_AND_TIME-Konstanten

DATE-Konstanten und TIME_OF_DAY-Konstanten können zu sogenannten DATE_AND_TIME-Konstanten kombiniert werden.

Syntax:

dt#<Datums- und Uhrzeitdeklaration>

Anstelle von "dt#" ist auch folgende Schreibweise gültig: "DT#", "date_and_time", "DATE_AND_TIME".

Die Datums- und Uhrzeitdeklaration muss im Format "<Jahr-Monat-Tag-Stunde:Minute:Sekunde>" angegeben werden.

Sekunden können dabei als reelle Zahlen angegeben werden, es können also auch Sekundenbruchteile angegeben werden.

DATE_AND_TIME (kurz DT) Werte werden intern wie DWORD behandelt. Die Zeit wird in Sekunden angegeben, wobei ab dem 1. Januar 1970 um 00:00 Uhr gerechnet wird.

Beispiele:

```
DATE_AND_TIME#1996-05-06-15:36:30
dt#1972-03-29-00:00:00
```

siehe auch: Zeitdatentypen

Zahlenkonstanten

Zahlenwerte, können als Dualzahlen, Oktalzahlen, Dezimal zahlen und Hexadezimalzahlen auftreten. Wenn ein Integerwert keine Dezimalzahl ist, dann muss seine Basis gefolgt von einem Doppelkreuz (#) vor die Integerkonstante geschrieben werden. Die Ziffernwerte für die Zahlen 10 bis 15 bei Hexadezimalzahlen werden wie üblich durch die Buchstaben A-F angegeben.

Unterstriche innerhalb eines Zahlenwertes sind erlaubt.

Beispiele:

14	(Dezimalzahl)
2#1001_0011	(Dualzahl)
8#67	(Oktalzahl)
16#A	(Hexadezimalzahl)

Der Typ dieser Zahlenwerte kann dabei BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL oder LREAL sein.

Implizite Konvertierungen von "größere" auf "kleinere" Typen sind nicht erlaubt. D.h. eine DINT- Variable kann nicht ohne weiteres als INT-Variable benutzt werden. Hierfür benutzt man die Typkonvertierungen.

REAL/LREAL Konstanten

REAL und LREAL-Konstanten können als Dezimalbrüche und in Exponentialdarstellung angegeben werden. Man verwendet hierbei die amerikanische Schreibweise mit Punkt.

Beispiele:

7.4	instead of 7,4
1.64e+009	instead of 1,64e+009

STRING-Konstanten

Ein String ist eine beliebige Zeichenreihe. STRING-Konstanten werden mit **einfachen Hochkommas** vorn und hinten begrenzt. Es können auch Leerzeichen und Umlaute eingegeben werden. Sie werden genauso wie alle anderen Zeichen behandelt.

In Zeichenfolgen wird die Kombination des Dollarzeichens ($) gefolgt von zwei hexadezimalen Ziffern als hexadezimale Darstellung des acht Bit Zeichencodes interpretiert. Außerdem werden, wenn sie in einer Zeichenfolge auftauchen, Kombinationen von Zeichen, die mit dem Dollarzeichen beginnen, wie folgt interpretiert:

angegebene Kombination	Interpretation
$<zwei hex.Zahlen>	hexadezimale Darstellung des 8Bit-Zeichen-Codes
$$	Dollarzeichen
$'	Einfaches Hochkomma
$L or $l	Zeilenvorschub
$N or $n	Neue Zeile
$P or $p	Seitenvorschub
$R or $r	Zeilenumbruch
$T or $t	Tabulator

Beispiele:

'w1Wüß?'

' Abby and Craig '

':-)'

'costs ($$)'

Getypte Konstanten / Typed Literals

Mit Ausnahme von REAL/LREAL-Konstanten (hier wird immer LREAL verwendet) wird beim Rechnen mit IEC-Konstanten der kleinstmögliche Datentyp verwendet. Soll ein anderer Datentyp verwendet werden, kann dies mithilfe von Typed Literals (Getypte Konstanten) erreicht werden, ohne dass die Konstante explizit deklariert werden muss. Die Konstante wird hierbei mit einem Prefix versehen, welches den Typ festlegt:

Syntax:

<Type>#<Literal>

<Type> gibt den gewünschten Datentyp an, mögliche Eingaben: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. Der Typ muss in Großbuchstaben geschrieben werden.

<Literal> gibt die Konstante an. Die Eingabe muss zum unter <Type> angegebenen Datentypen passen.

Beispiel:

```
var1:=DINT#34;
```

Wenn die Konstante nicht ohne Datenverlust in den Zieltyp überführt werden kann, so wird eine Fehlermeldung ausgegeben:

Getypte Konstanten können überall dort verwendet werden, wo normale Konstanten verwendet werden können.

16.4.2 Variablen

Variablen werden entweder lokal im Deklarationsteil eines Bausteins deklariert oder in den globalen Variablenlisten.

Siehe Variablendeklaration für Informationen zur Deklaration einer Variable und zur Mehrfachverwendung von Variablenbezeichnern.

Wo eine Variable verwendet werden kann, hängt von ihrem Datentyp ab.

Die verfügbaren Variablen können über die Eingabehilfe abgerufen werden.

Zugriff auf Variablen von Arrays, Strukturen und Bausteinen

Verwenden Sie folgende Schreibweise für den Zugriff auf ...

Zweidimensionale Array-Komponenten:

<Array-Name>[Index1, Index2]

Strukturvariablen:

<Strukturname>.<Variablenname>

Funktionsblock- und Programmvariablen:

<Funktionsblockname>.<Variablename>

Bitzugriff auf Variablen

In ganzzahligen Variablen können einzelne Bits angesprochen werden. Dazu wird an die Variable mit einem Punkt abgetrennt der Index des zu adressierenden Bits angehängt. Der Bit-Index kann durch eine beliebige Konstante gegeben werden. Die Indizierung ist 0-basiert.

Syntax:

<Variablename>.<Bitindex>

Beispiel:

```
a : INT;
b : BOOL;
...
a.2 := b;
```

Das dritte Bit der Variablen a wird auf den Wert der Variable b gesetzt.

Ist der Index größer als die Bit-Breite der Variablen so wird folgender Fehler ausgegeben: Index '<n>' außerhalb des gültigen Bereichs für Variable '<var>'!

Die Bitadressierung ist bei folgenden Variablentypen möglich: SINT, INT, DINT, USINT, UINT, UDINT, BYTE, WORD, DWORD.

Ist der Typ der Variablen nicht zulässig, so wird folgende Fehlermeldung ausgegeben: Unzulässiger Datentyp '<Typ>' für direkte Indizierung".

Ein Bit-Zugriff darf nicht einer VAR_IN_OUT-Variablen zugewiesen werden!

Bitzugriff mit Hilfe einer globalen Konstante:

Wenn eine globale Konstante deklariert wird, die die Bitnummer definiert, kann diese Konstante für den Bitzugriff verwendet werden

Beispiele für Bitzugriff über eine globale Konstante...

1. auf eine Variable und 2. auf eine Strukturvariable:

Deklaration in einer globalen Variablenliste für beide Beispiele:

Variable *enable* definiert, auf welches Bit zugegriffen werden soll:

```
VAR_GLOBAL CONSTANT
enable:int:=2;
END_VAR
```

Beispiel 1, Bitzugriff auf eine ganzzahlige Variable:

Deklaration in Baustein:

```
VAR
xxx:int;
END_VAR
```

Bitzugriff:

```
xxx.enable:=true;   (* -> das 3.Bit in Variable xxx wird TRUE gesetzt *)
```

Beispiel2, Bit-Zugriff auf ganzzahlige Strukturkomponente:

Deklaration der Struktur *stru1*:

```
TYPE stru1 :
STRUCT
bvar:BOOL;
rvar:REAL;
wvar:WORD;
{bitaccess enable 42 'Start drive'}
END_STRUCT
END_TYPE
```

Deklaration in Baustein:

```
VAR
x:stru1;
END_VAR
```

Bitzugriff:

```
x.wvar.enable:=true;
```

Dadurch wird das 42. Bit in Variable x auf TRUE gesetzt. Da bvar 8 Bit und rvar 32 Bit enthalten, erfolgt dieser Zugriff auf das 2.Bit in der Variable wvar, die somit den Wert 4 erhält.

16.4.3 Adressen

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen !

Adresse

Bei der Angabe einer Adresse werden die Position im Speicher und die Größe mittels spezieller Zeichenfolgen ausgedrückt.

Syntax:

%<Bereichspräfix><Größenpräfix><number|.number|.number....>

Folgende Speicherbereichspräfixe werden unterstützt:

I	Eingangsspeicherbereich ("Inputs", physikalische Eingänge über Eingangstreiber, "Sensoren")
Q	Ausgangsspeicherbereich ("Outputs", physikalische Ausgänge über Ausgangstreiber, "Aktoren")
M	Merkerspeicherbereich

Folgende Präfixe für das Format (Größe) werden unterstützt:

X	Single bit
None	Single bit
B	Byte (8 Bits)
W	Word (16 Bits)
D	Double word (32 Bits)

Beispiele:

%QX7.5 and %Q7.5	Ausgangsbit 7.5
%IW215	Eingangswort 215
%QB7	Ausgangsbyte 7
%MD48	Doppelwort an Speicherstelle 48 im Merkerbereich.
%IW2.5.7.1	Interpretation abhängig von der aktuellen Steuerungskonfiguration (siehe unten)
ivar AT %IW0 : WORD;	Beispiel einer Variablendeklaration mit Adressangabe

Sicherstellen, dass die Adresse gültig ist:

Um in einer Applikation eine gültige Adresse zuzuweisen, muss zunächst die geforderte Position im Prozessabbild bekannt sein; d.h. der zuständige **Speicherbereich**: Eingangs- (I), Ausgangs- (Q) oder Merkerspeicherbereich (M), siehe oben. Weiterhin muss die erforderliche **Größe** angegeben werden: Bit, Byte, Word, DWord (siehe oben: X, B, W, D)

Eine entscheidende Rolle spielt die **aktuell verwendeten Gerätekonfiguration** und ihren Einstellungen ab (Hardware-Struktur, Gerätebeschreibung, E/A-Einstellungen). Zu beachten sind vor allem die Unterschiede der Adress-Interpretation bei Geräten mit "Byte-Adressierungsmodus" und solchen mit word-orientiertem "IEC-Adressierungsmodus".

Abhängig von der Größe und dem Adressierungsmodus können also mit der gleichen Adressangabe unterschiedliche Speicherzellen angesprochen werden.

Die nachfolgende Tabelle zeigt die Gegenüberstellung der **Byte-Adressierung** und **Word-orientierten IEC-Adressierung** für Bits, Bytes, Words und DWords. Auch zeigt sie die **überlappenden Speicherbereiche**, die im Falle der Byte-Adressierung vorliegen (siehe auch das Beispiel im Anschluss an die Tabelle).

Bezüglich der Schreibweise ist zu beachten, dass der IEC-Adressierungsmodus immer word-orientiert ist, d.h. dass die vor dem Punkt die Word-Nummer steht, danach die Bit-Nummer.

DWords / **W**ords				**B**ytes	**X** (bits)					
byte addressing		word oriented IEC addressing			byte addressing			word oriented IEC addressing		
D0	W0	D0	W0	B0	X0.7	...	X0.0	X0.7	...	X0.0
D1	W1			B1	X1.7	...	X1.0	X0.15	...	X0.8
...	W2		W1	B2	...			X1.7	...	X1.0
	W3			B3				X1.15	...	X1.8
	W4	D1	W2	B4						
	...			B5						
			W3	B6						
				B7						
		D2	W4	B8						
		...		...						
		...		...						
		...		...						
D(n-3)		D(n/4)	...							
	W(n-1)		W(n/2)							
				B n	Xn.7	...	Xn.0	X(n/2).15	...	X(n/2).8

Abb. 540 Gegenüberstellung von byte- oder wort-orienterter Adressierung der Adressformate D, W, B und X :

n = Byte-Nummer; word oriented = wortorientiert, addressing = Adressierung

Speicherbereich-Überlappung im Falle des Byte-Adressierungsmodus, Beispiel:

D0 enthält B0 - B3, W0 enthält B0 und B1, W1 enthält B1 und B2, W2 enthält B2 und B3 -> Um die Überlappung zu umgehen, dürfen W1 bzw. D1, D2, D3 nicht für die Adressierung verwendet werden!

Hinweis: Boolsche Werte werden byteweise alloziert, wenn die nicht explizit eine Einzelbitadresse angegeben wird. Beispiel: Eine Wertänderung von varbool1 AT %QW0 betrifft den Bereich von QX0.0 bis QX0.7.

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen !

16.4.4 Funktionen

In ST kann auch ein Funktionsaufruf als Operand auftreten.

Beispiel:

```
Result := Fct(7) + 3;
```

TIME()-Funktion

Diese Funktion liefert die Zeit auf Millisekunden-Basis, die seit Systemstart vergangen ist.

Der Datentyp ist TIME.

Beispiel in AWL:

```
TIME
ST systime  (* Ergebnis z.B.: T#35m11s342ms *)
Example in ST:
systime:=TIME();
```

17 Geräte

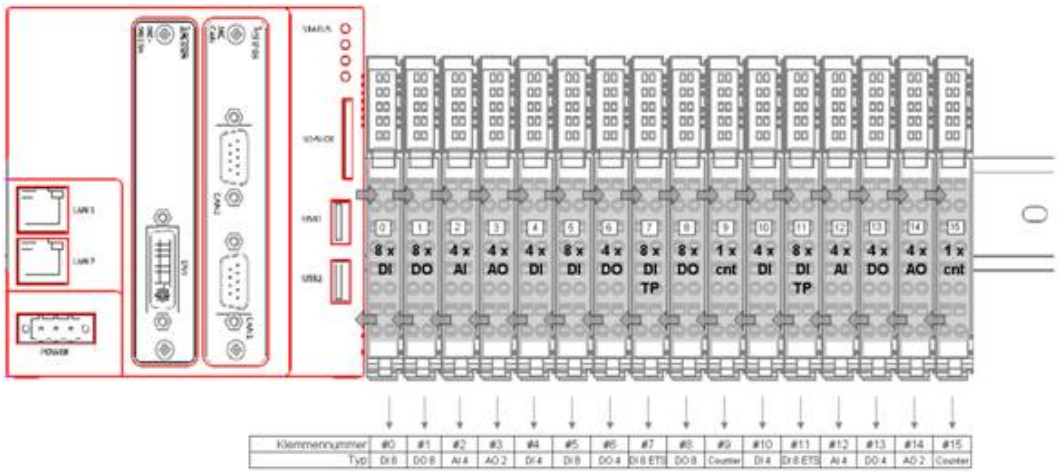
Hinweis zur Verwendung einer Hilscher cifX Karte

Zur Verwendung einer cifX Karte von Hilscher GmbH (mit entsprechendem IEC-Biblithek als Treiber) muss die »PLC Designer« Konfigurationsdatei (.cfg) um folgenden EIntrag ergänzt werden:

Component.<nächste freie Nummer>=CmpHilscherCIFX

17.1 L-force Controller 3200 C

Die folgende Abbildung zeigt schematisch den Hardware-Aufbau eines Motion-Systems.



Das Motion-System besteht aus den Komponenten:

- L-force Controller 3221 C oder 3231 C mit einer SPS-Steuerung in Form einer Soft-PLC.
- I/O-Komplettmodule, die über einen internen Rückwandbus mit dem L-force Controller Prozessdaten, Parameterdaten und Diagnosinformationen austauschen

Um in einem SPS-Programm auf die I/O-Komplettmodule zuzugreifen, d. h. Eingangssignale zu lesen und Ausgänge zu schreiben, ist die Erstellung einer Klemmenkonfiguration im PLC Designer notwendig.

Hierzu muss die Anordnung der I/O-Komplettmodule in der Steuerungskonfiguration der PLC nachgebildet werden.

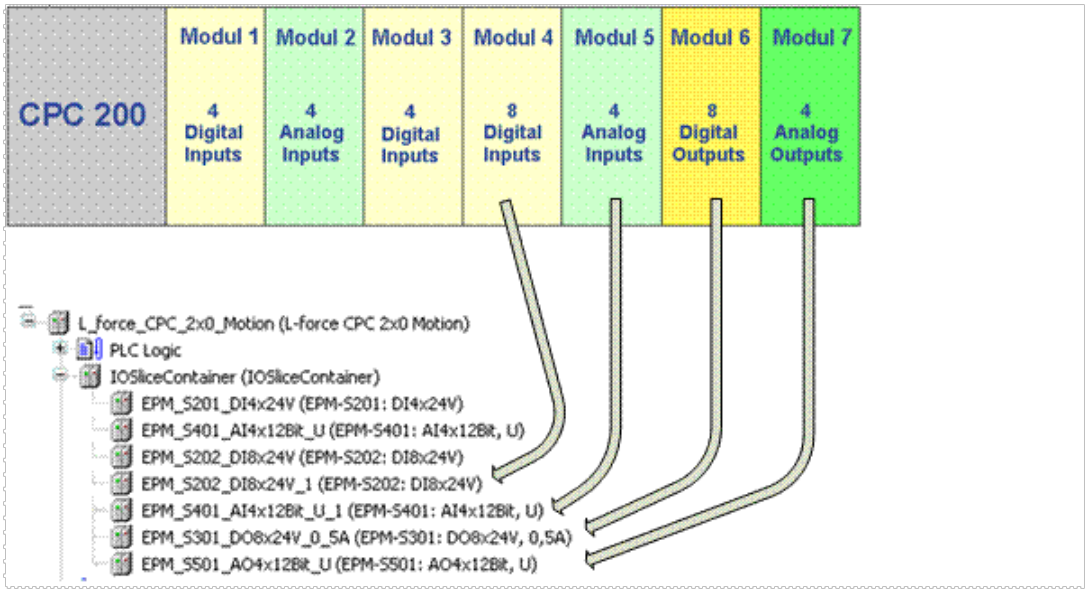
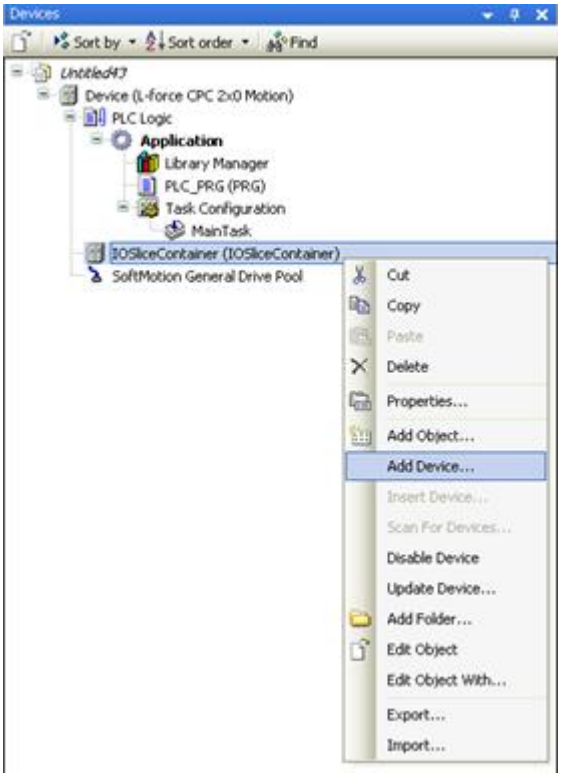


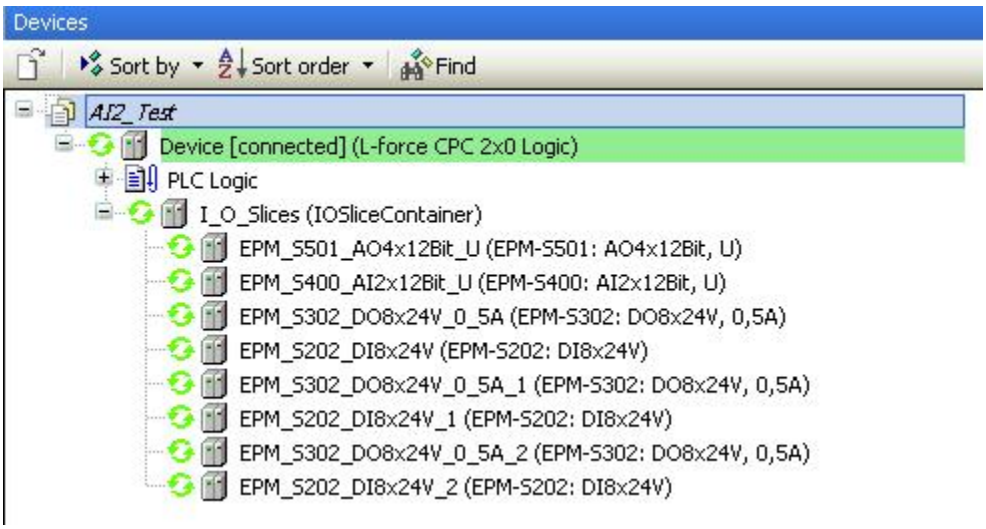
Abb. 541 Beispiel: Abbildung der physikalischen Topologie in der Steuerungskonfiguration

Die Konfiguration erfolgt, indem unterhalb des L-force Controllers im *IOSliceContainer* die I/O-Komplettmodule als entsprechende Devices eingefügt werden:



Wichtig ist, dass die Reihenfolge der I/O-Komplettmodule im Gerätebaum mit der physikalischen Anordnung übereinstimmt.

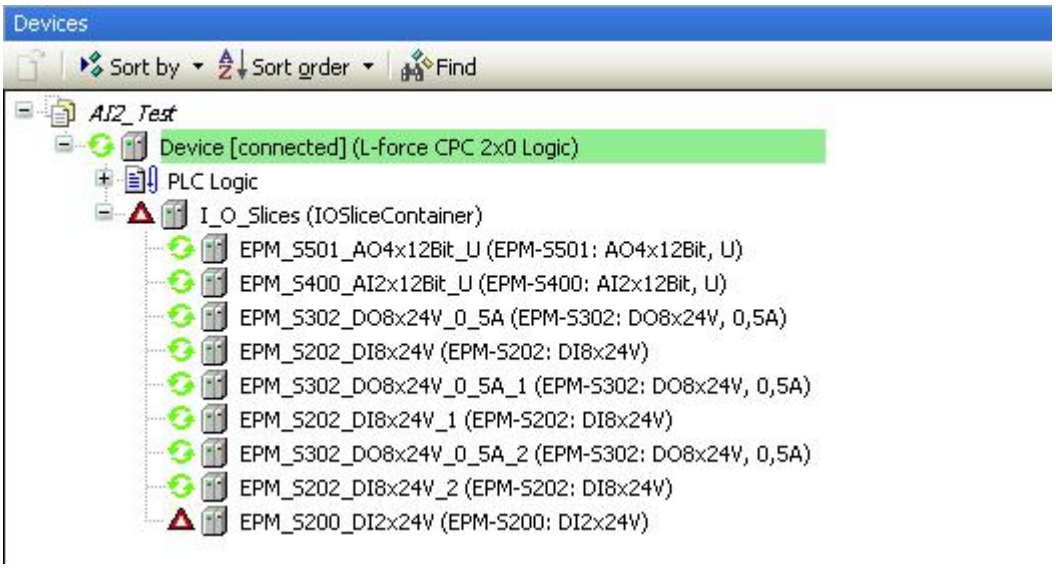
Ist alles richtig konfiguriert, sind nach dem "Online gehen" und Starten der PLC alle I/O-Komplettmodule mit einem grünen Kreis im Gerätebaum gekennzeichnet.

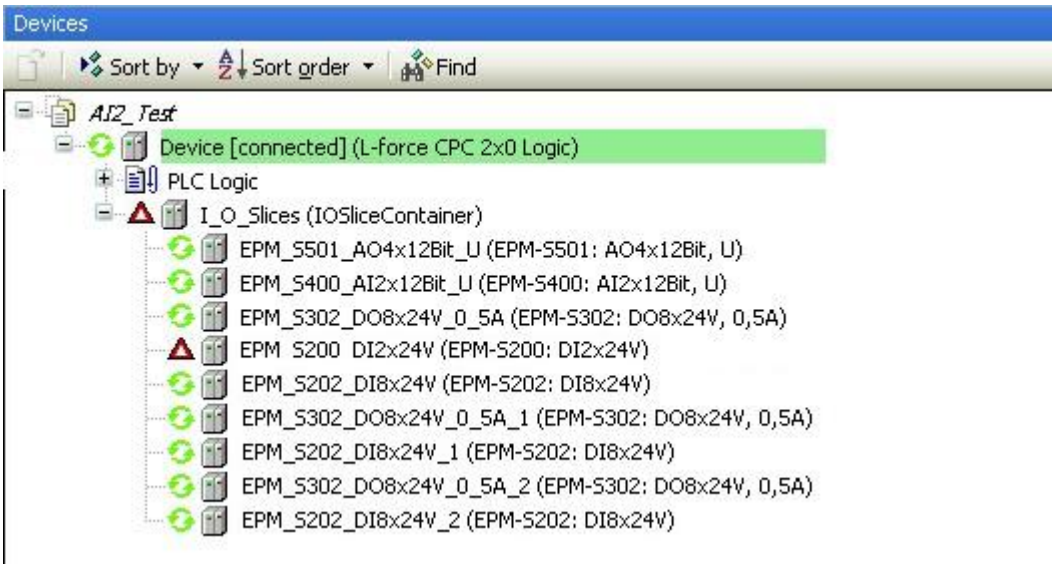


Im Fehlerfall, d. h. die Konfiguration passt nicht zur physikalischen Anordnung, werden die I/O-Komplettmodule mit einem roten Dreieck gekennzeichnet.

Beispiele fehlerhafter Konfigurationen:

EPM_S200 ist in der physikalischen Konfiguration nicht vorhanden.

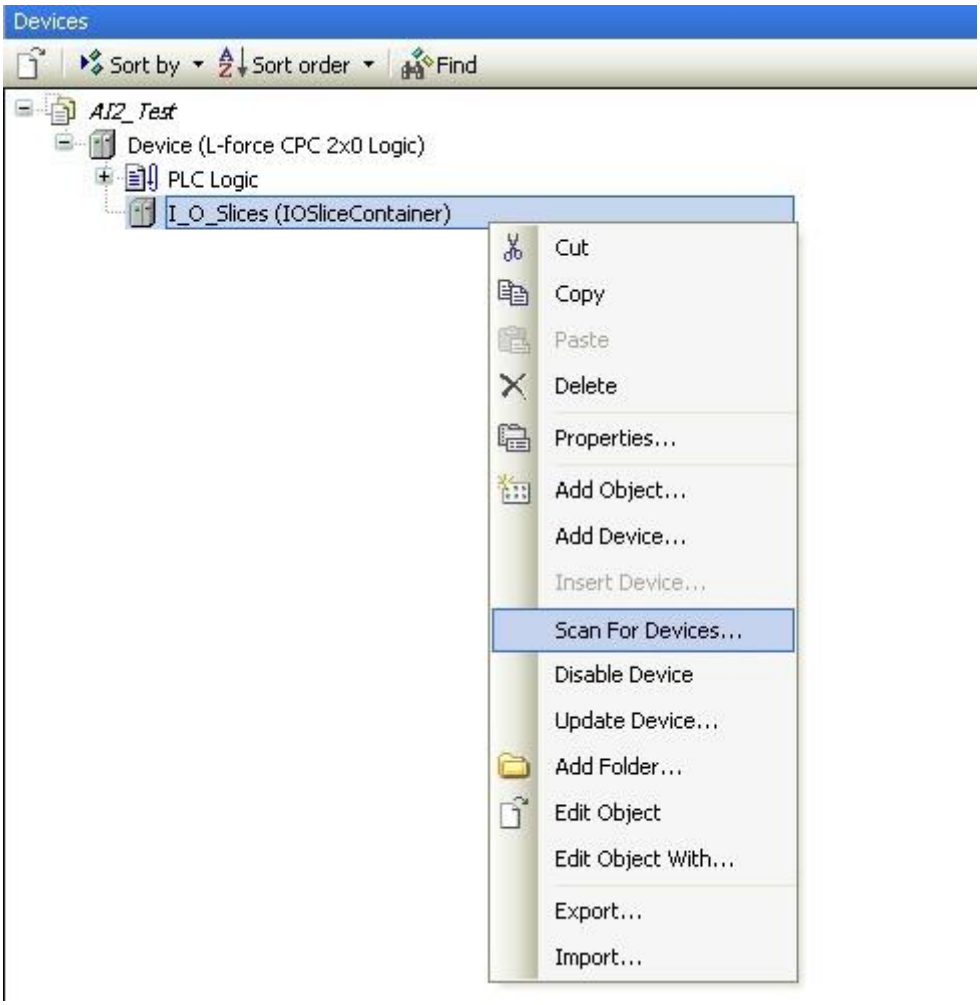




17.1.1 Automatisches Erkennen der I/O-Komplettmodul-Topologie

Neben der manuellen Konfiguration ist auch ein automatisches Erkennen der I/O-Komplettmodule im PLC Designer-Projektbaum möglich.

Die Funktion wird über den Menüpunkt "Scan For Devices" im PLC Designer ausgewählt:



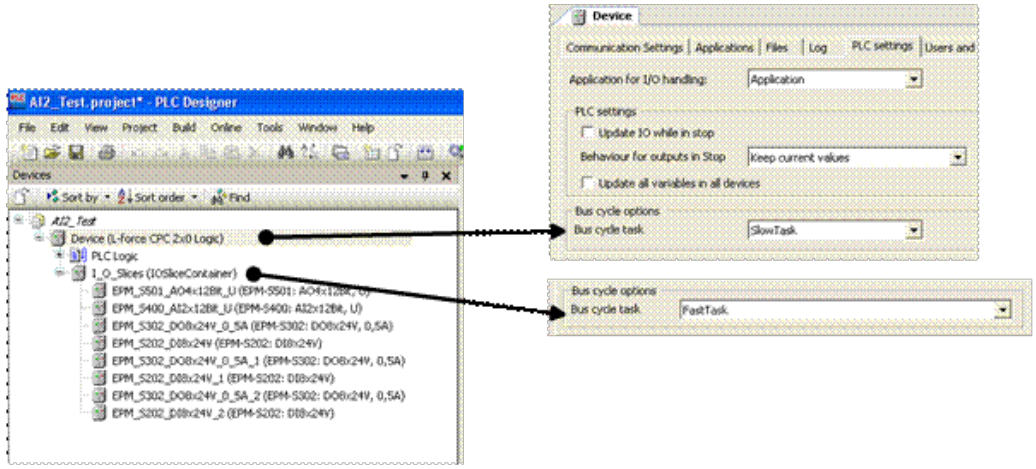
17.1.2 Zykluszeit des Zugriffs auf die I/O-Komplettmodule einstellen

Im Device "L-force Controller" wird zentral eine Buszyklustask für die Bearbeitung aller angeschlossenen Feldbusse sowie für die über den Rückwandbus angeschlossenen I/O-Komplettmodule ausgewählt.

Diese Auswahl stellt eine Vorbelegung dar und kann in den Busmaster Knoten (EtherCAT Master, CANopen_Manager, IOSliceContainer, ...) überschrieben werden.

Soll der Rückwandbus mit einer kürzeren Zykluszeit als im Device "L-force Controller" ausgewählt betrieben werden, so kann im ‚IOSliceContainer' eine schnellere Task ausgewählt werden.

Beispiel: Unterschiedliche Zykluszeiten



Im PLC Designer wird für die SPS zunächst eine Bearbeitungstask für die Bildung des I/O-Abbildes festgelegt. Davon jedoch abweichend kann für jedes einzelne I/O-Komplettmodule eine eigene sogenannte Buszyklus-Task bestimmt werden, in der dann das Prozessabbild für diese I/O-Komplettmodule erstellt wird.

Insgesamt können drei unterschiedliche Task für die I/O-Komplettmodul-Bearbeitung definiert werden. Die Beschränkung auf drei Task ist systembedingt durch die Rückwandbusspezifikation vorgegeben. Der Rückwandbus-Controller verwendet für jede Task zwei interne Buszyklusgruppen. Eine für Eingangssignale, sowie eine für die Ausgänge. Hierbei stehen insgesamt 6 Gruppen zur Verfügung.

17.1.3 Rechenzeitbedarf einzelner SoftMotion Funktionsbausteine

Funktionsbaustein:	Decoder (no Splines)	Decoder (splines + Checkvelocities)	SmoothPath + CheckVelocities	Interpolator (no splines, trapezoid)
L-force Controller 3221 C	13 µs	306 µs	152 µs	19 µs
L-force Controller 3231 C	12 µs	260 µs	116 µs	17 µs

Funktionsbaustein:	Interpolator (no splines, sigmoid)	Interpolator (mixed)	Interpolator (splines)	Check-Velocities
L-force Controller 3221 C	21 µs	54 µs	241 µs	130 µs
L-force Controller 3231 C	18 µs	44 µs	183 µs	97 µs

Funktionsbaustein:	Camming XYVA Tables	Camming equidistant tables	Gearing	Phasing
L-force Controller 3221 C	15 µs	18 µs	13 µs	15 µs
L-force Controller 3231 C	15 µs	18 µs	12 µs	15 µs

Funktionsbaustein:	MoveVelocity	Stop	MoveAbsolut	PositionProfile
L-force Controller 3242 C	15 µs	16 µs	16 µs	14 µs
L-force Controller 3221 C / 3232 C	13 µs	15 µs	15 µs	13 µs

18 PLC-Designer Bibliotheken

Es hängt vom verwendeten Zielsystem ab, welche der folgenden Bibliotheken im Steuerungsprogramm verwendet werden können:

- **Bibliothek Standard.Library**
- **Bibliothek Util.library**
- **Busspezifische Bibliotheken:**
  - Bibliothek IIoDrvBusControl.Library
  - Bibliothek IIoDrvDPV1C1.library
  - Bibliothek IIoDrvDPV1C2.library
  - Bibliothek IIoDrvProfibus.Library
  - Bibliothek IIoDrvAcyclicServices.library
  - Bibliothek CAA_CiA405.lib
- **Bibliothek SM3_Basic.Library**
- **Bibliothek SM3_CNC.Bibliothek**
- **Systembibliotheken:**
  - Runtime Error Codes
  - SysCom.library
  - SysComAsync.library
  - SysCPUHandling
  - SysDir.library
  - SysDirAsync.library
  - SysEvent.library
  - SysFile.library
  - SysFileAsync.library
  - SysInt.library
  - CoDeSys V3.x System Libraries
  - SysMem.library
  - SysPci.library
  - SysPort.library
  - SysPortAsync.library
  - SysProcess.library
  - SysSem.library

  - SysSemProcess.library
  - SysShm.library
  - SysSocket.library
  - SysSocketAsync.library
  - SysTask.library
  - SysTime.library
  - SysTimer.library
  - SysTypes.library
- **CAA Bibliotheken:**
  - CAA_AsyncManager.library
  - CAA_CallBack.Library
  - CAA_CanL2.library
  - CAA_CanOpenMan.library
  - CAA_CanOpenStack.library
  - CAA_CiA405.library
  - CAA_DTUtil.library
  - CAA_FILE.library
  - CAA_Mathematics.library
  - CAA_MemBlockMan.library
  - CAA_Memory.library
  - CAA_NetBaseServices.library
  - CAA_RealTimeClock.library
  - CAA_ResMan.library
  - CAA_SegBufferMan.library
  - CAA_SerialCom.library
  - CAA_Settings.library
  - CAA_Tick.library
  - CAA_Timer.library
  - CAA_Types.library

Hinweis: Weitere Informationen zu den Bibliotheken finden Sie in der Online-Hilfe.

19 Index

__DELETE 1148, 1150
__ISVALIDREF 1092, 1106, 1150
__NEW 1148, 1150
__QUERYINTERFACE 1153
__QUERYPOINTER 1155
1006h Objekt 934
2. y-Achse aktivieren 825
3D-Modus 655
Abarbeitungsreihenfolge in AS 790
Abarbeitungsreihenfolge von Tasks 818
Abbruch bei Fehler 943
Abgerundetes Rechteck 839
Ablaufsprache 766
Abmelden-Dialog 545
Abrufen 516
Abrufen automatisch 473
Abrufen über Projektgeschichte-Dialog 520
ABS 1142
absolut 588, 598
absolute Adresse 120
Absolute Bewegung von Visualisierungselementen 843
absoluter Wert 1142
Achsen für Trace-Aufzeichnung 825
Achsenbegrenzungen 993
Achsenbeschriftung für Trace-Aufzeichnung 825
Achsentyp 993
Achsfehler SoftMotion 54
Achssteuerung
  Beispielapplikation 1002, 1005
  Beispielapplikationen 1001
Achssteuerung 1001
Achssteuerung 1005
Achssteuerung mit Kurvenscheibe 1006
ACOS 1146
ADD 1112
Addition 1112
Address Operator ADR 1093, 1107
ADR 1093, 1107, 1131, 1157
Adresse
  absolut 120
  ändern 911
  Bestimmung 118
  Broadcast 120
  fixieren 911
  FUP
    KOP und AWL Editor 461
  Gerät 911
  manuell ändern 911
  relativ 120
Adresse 497
Adresse 1037
Adresse 1169
Adresse 1169
Adresse auflösen 901
Adresse eines E/A-Kanals 911
Adresse Gerät 287
Adresse in AWL 720
Adressierung 118, 120
Adressierung und Routing 118, 120
Adressoperator 1131, 1157
Adressoperator ADR 1093, 1107
Adress-Präfix 1169
Adressstrukturen 120
Adresszuweisung 1049
Aktion
  Aufruf 90
  Hinzufügen 90
Aktion 90
Aktion ausführen 560
Aktion in AS 505, 771, 772
Aktionen
  Zugriffsrechte 921
Aktionen 546

Aktionen auf Gerät erlauben 921
Aktionsassoziation 505
Aktionsassoziation danach einfügen 505
Aktionsassoziation einfügen 505
Aktionsaufruf 86
Aktionsaufruf in AWL 720
Aktionsbreite 467
Aktionsname 772, 1028
Aktions-Qualifizierer 505, 772, 783
Aktionsstatus 784
Aktionstyp 888
Aktiv
  Watchdog 814
Aktivationssstatus 784
Aktive Applikation 412
Aktive Applikation setzen 412
Aktive Randfarbe 445
Aktiven Pfad setzen 306
Aktiver Editor 329
Aktivieren eines Haltepunkts 557
Aktivieren Watchdog 814
Aktualisieren
  Applikationen-Liste 905
Aktualisieren der Installation 146
Aktualisieren von Versionen 338
Aktualisierung Bausteinparameter in FUP/KOP 310
Aktualisierung von Parametern in FUP/KOP/AWL 325
Aktualisierungsrate Web-Browser 896
Aktualsieren der Versionen 365
Aktuelle Trefferanzahl 557
Aktuelle Zeile 445
Aktuelle Zeile hervorheben 445
aktueller Benutzer 545
Akualisierungsintervall Data Server 697
Alarmfarbe 843
ALIAS 1092, 1106
Alle aktivieren 557
Alle Ausgänge auf Standardwert 909
Alle deaktivieren 557
Alle Editoren schließen 586
Alle Forces betrachten 898
Alle Forces zum Überwachen-Fenster hinzufügen 435
Alle löschen 557
Alle SDOs erzeugen 936
Alle Variablen in allen Geräten aktualisieren 909
Alles auswählen 281
Alles bereinigen 252
Alles deselektieren 549
Alles selektieren 549
Allgemeine Eigenschaften 561
Als Funktionsbausteinsprache anzeigen 326
Als Kontaktplan anzeigen 327
Als Standard verwenden 825
Alternativ 503
Alternative Verzweigung 503, 772
Altes Passwort 378
An den Anfang 260
AND 1117
AND Operator in AWL 713
Ändern von Variablenwerten 659
ANDN 1118, 1159
Andocken 166, 585
Anemldename 378
Angegebene Client-Größe verwenden 895, 896
Anhalten
  wenn... 241
Anmelden 525
Anmelden bei Quellcodeverwaltung 525
Anmelden-Dialog 543
Anordnung von Visualisierungselementen 842
Anpassen
  Menü 479
  Symbolleisten 477
  Tastatur 482
Anpassen im Online Modus 474
Ans Ende 260

Anschlüsse zurücksetzen 267
Ansicht
  Geräte 560, 561
  Meldungen 580
  POUs 64, 560
  Quellcodeverwaltung 581
  Überwachen 584
  Überwachungsliste 584
  Werkzeuge 583
Ansicht 166
Ansicht 560
Ansicht ausblenden 166
Ansicht Befehle
  messages 580
Ansicht Befehle 557, 561
Ansicht Befehle 581
Ansicht Befehle 583
Ansicht im Online-Modus 169
Ansicht in FUP/KOP/AWL 327
Ansicht in FUP/KOP_AWL 326, 327
Ansicht Menü 171
Ansicht schließen 166
Ansicht-Menü 166
Ansicht-Optionen AS-Editor 467
Anstehende Eincheckvorgänge 582
Antrieb
  Beispielapplikation 1002
  frei 992
  gekoppelt 992
  KOnfiguration 993
Antrieb ausschalten 999
Antrieb einschalten 999
Antrieb initialisieren 999
Antriebskonfiguration 989, 993
Antriebssteuerung
  Beispielapplikationen 1001
Antriebssteuerung 999
Anweisung 712, 799
Anweisungsliste 712, 720
Anweisungslisten-Ansicht 327
Anwenderparameter DP-Slave 982
Anwenderparameter für DP-Modul 985
ANY_..._TO Konvertierungen 1142
ANY_NUM_TO 1142
ANY_TO 1142
ANY_TYPE 1091
Anzahl der Syncs 939
Anzahl IEC-Zyklen 818
Anzahl Objekte 366
Anzahl von Rückgängig-Schritten 445
Anzahl Zyklen 818
Anzeige Objektversionsgeschichte 522
Anzeige zurücksetzen 539
Anzeigeeinstellungen 825, 834
Anzeigeeinstellungen im Trace als Standard verwenden 825
Anzeigeeinstellungen rücksetzen 825
Anzeigen über Projektgeschichte-Dialog 520
app-Datei 134
Applikation
  aktiv 412
  Anordnung 108
  Code-Generierung 249
  Details 134
  Haltepunkt-Position 557
  Hierarchie 108
  Login-Dialog 134
  Name 108
  Online Change 134
  Übersetzen 248
  Vater und Kind 108
Applikation 114
Applikation bereinigen 252
Applikation für E/A-Behandlung 909
Applikation für I/O-Handling 911
Applikation für Symbole 232

Applikation laden 424
Applikation neu übersetzen 249
Applikation starten 427
Applikation stoppen 427
Applikation übersetzen 248
Applikation zurücksetzen 430, 431
Applikationen auf Steuerung 905
Applikationen löschen 441, 905
Applikationsbeispiele SoftMotion 1001
Applikationshierarchie 114
Applikationsinformation 134
Arbeitsblatt bearbeiten 256
Arbeitsblatt in CFC 621
Arbeitsblattgröße CFC 256
Archiv
  Archiv extrahieren 352
  Projekt archivieren 349
  Projektarchiv 352
Archiv 349
Archiv 352
Archiv extrahieren 352
Archiv speichern 349
Archiv versenden 349
archive.prj 425, 426
Arcus Cosinus 1146
Arcus Sinus 1146
Arcus Tangens 1147
Array
  Deklaration 497
  Pragma 1060
  Tabelle in Visualisierung 843
  Visualisierung 1060
Array 1167
ARRAY
  Deklaration 1097
  Initialisierung 1097
ARRAY 1097
Array einer Struktur 1097
Array in Visualisierung 213
Array-Assistent 497
AS
  Abarbeitungsreihenfolge 790
  Cursor-Positionen 767
  Flags 784
  implizite Variablen 561
  Implizite Variablen 784
  Online-Modus 792
  Qualifizierer 783
  Schrittattribute 771
AS 766
AS Attribut 467
AS Bibliothek 561
AS Eigenschaften 470
AS Elemente 772
AS Elementeigenschaften 581
AS Flags 561
AS Makro 507, 508
AS Optionen 470
AS Schritt 501
AS Sprung 506
AS Standardeinstellungen 470
AS Werkzeugbox 772
AS-Aktion 505
AS-Bibliothek 470, 783, 784
ASCII 253
AS-Editor
  Bearbeiten 768
  Optionen 467
AS-Editor Befehle 500
AS-Editor Optionen 467
AS-Einstellungen 561
AS-Flags 784
AS-Flags Optionen 470
ASIN 1146
AS-Menü 500
AS-Optionen 561

AS-Schritt
Attribute 771
Zeitüberwachung 771
Assign other names 1092, 1106
Assoziation für AS-Aktion 505
asynchrone PDOs 939
AT 1025, 1037
ATAN 1147
AT-Deklaration 1037
Attribut 1070, 1072, 1073
Attribut displaymode 1059
Attribut ExpandFully 1060
Attribut Global_init_slot 1061
Attribut Hide 1062
Attribut Hide_all_locals 1063
Attribut Init_Namespace 1064
Attribut Init_On_Onlchange 1065
Attribut Initialize_on_call 1064
Attribut Instance-path 1065
Attribut linkalways 1066
Attribut Monitoring 1067
Attribut No_check 1068
Attribut No_Copy 1068
Attribut No_virtual_actions 1070
Attribut No-exit 1069
Attribut Noinit 1069
Attribut Obsolete 1072
Attribut 'parameterstringof' 884
Attribut Parameterstringof 1074
Attribut Qualified_Only 1075
Attribut Reflection 1076
Attribut Symbol 1076
Attributbreite 467
Attribute eines AS-Schritts 771
Attribute für Variablentypen 1045
Attribut-Pragmas
Attribut Hide 1062
Attribut Monitoring 1067
Attribut No-exit 1069
Attribut Qualified_Only 1075
Attribut Reflection 1076
Benutzerdefinierte Attribute 1057
displaymode 1059
ExpandFully 1060
Global_init_slot 1061
Hide_all_locals 1063
Init_Namespace 1064
Init_On_Onlchange 1065
Initialize_on_call 1064
Instance-path 1065
linkalways 1066
No_check 1068
No_Copy 1068
no_virtual_actions 1070
Noinit 1069
Obsolete 1072
Pack_mode 1073
Parameterstringof 1074
Symbol 1076
Attribut-Pragmas 1056
Auf Aktualisierungen hin überprüfen
wenn dieses Projekt geladen wird 365
auf das Gerät laden 917, 921
auf Gerät laden 921
auf Platte speichern 917, 921
Aufruf einer Methode 80, 84
Aufruf einer POU 66
Aufruf eines Funktionsblocks 75
Aufruf Programm 68
Aufrufliste 559
Aufrufoperator 1132
Aufrufparameter 75
Aufzählung 1101
Aufzeichnen in jedem x-ten Zykluns 825
Aufzeichnung
Darstellung 825

Aufzeichnung 820
Aufzeichnung für Trace 825
Aufzeichnungsbedingung für Trace-Aufzeichnung 825
Aufzeichnungseinstellungen 820
Ausblenden 166, 585
Auschecken automatisch 473
Auschecken rückgängig 519, 582
Auschecken... 516
Ausdruck
  Auswertung 798
Ausdruck 484, 708
Ausdruck 798
Ausdruck überwachen 897
Ausdrucken von Projektinhalt 493
Ausführen bis Cursor 245
Ausführen bis Rücksprung 245
Ausführungsreihenfolge in CFC 260, 261, 264, 265, 621
Ausgabeadresse 1169
Ausgabevariablen 1042
Ausgang in CFC 616
Ausgänge auf Default setzen 909
Ausgänge bei Gerätestop 909
Ausgänge bei Stop 909
Ausgänge EtherCAT Master 954
Ausgänge EtherCAT Slave 959, 965
Ausgangsaktion 771, 772
Ausgangsdaten Adresse 951
Ausgangskanal 911
Ausgangslänge 985
Ausgangsspeicher 1169
Ausgangsvariablen 1042
Ausgecheckt lassen 518
ausgewählte Visualisierungen 550
Ausgewähltes Gerät löschen 305
Ausloggen 422
Ausloggen aus Applikation 422
Ausrichtung Papier 485
Ausrichtung Visualisierungselement 842
Ausrichtung von Text in Visualisierung 187
Ausrichtung von Visualisierungselementen 553, 554
Ausschneiden 279
Ausschneiden in AS 768
Ausschneiden in AWL 720
Aussehen Visualisierungselement 843
Aussehen-Variablen für Visualisierungselement 843
Außerhalb des Anzeigebereichs 538, 541
Auswahl von Visualisierungselementen 219, 554
Auswählen
  Visualisierungselemente 549
Auswählen 679
Auswählen in AS 767, 768
Auswählen von Visualisierungselementen 841
Auswahloperator 1124
auto clear mode 978
Auto-Clear-Modus 978
Autodeklaration 452, 497, 1035
Autoformat 452
AutoInc Adresse 955
Automatisch berechnete Client-Größe verwenden 895
Automatisch einrücken 445
Automatisch POUs für Eigenschaftenzugriff 366
Automatisch scrollen in Trace-Aufzeichnung 825
Automatisch skalieren in Trace-Aufzeichnung 825
Automatisch speichern 443
Automatisch Speichern 338
automatisch starten 978
Automatische Anpassung im Online-Modus 895, 896
Automatische Anpassung Web-Visualisierung 896
Automatische Konfiguration Master/Slaves 951
Automatisches Mapping 935
Automatisches Verknüpfen aktivieren 459
Autor 366
Autosave 443
autosave-Datei 443
AWL 712

AWL Befehle 730
AWL Editor Befehle 307
AWL Editor Optionen 461
AWL Modifikator 713
AWL Operand 720
AWL operator 713, 720
AWL Postfix 720
AWL Präfix 720
AWL Tabelleneditor 720
AWL-Ansicht 327
AWL-Editor 720
AWL-Zeile löschen 321
AXIS_REF_LOGICAL_SM3 997
azyklische PDOs 939
azyklische Übertragung 939
B 1169
backup-Datei 443
Backup-Datei 338
Bahnansicht scrollen 270
Bar Display 204
batch mode
  project 354
Batch-Modus
  file close 345
  file merge 354
  file new 335
  file save 346
  file saveas 347
Baudrate beim Profibus 978
Baudrate für CANbus 924, 930
Baustein in CFC 616
Baustein in FUP/KOP/AWL 735
Bausteinaufruf einfügen 310
Bausteinaufruf in FUP/KOP/AWL 310
Bausteineingang einfügen 316
Bausteineingang in FBD/LD 316
Bausteinindex 1117, 1158
Bausteinname 1028
Bausteinparameter aktualisieren 325
Bausteinsymbol anzeigen in FUP/KOP 461
Bearbeiten Menü 171
Bearbeitungsmodus 526
Bedienfeld 677
bedingte Kompilation 1078
bedingte Kompilierung 561
bedingte Pragmas 1078
bedingte Übersetzung 1078
bedingter Aufruf 713
bedingter RETURN 713
bedingter Sprung 713
Bedingung
  Haltepunkt 557
Bedingung für Haltepunkt 241
Beenden 358
Befehle
  Übersicht 171
Befehle der Kategorie Drucken 483
Befehle der Kategorie Geräte
  EtherCAT XML erzeugen 285
Befehle der Kategorie Haltepunkte 240
Befehle der Kategorie Visualisierungskommandos 548
Befehle für Lesezeichen 238
Befehlskategorie
  Ansicht 554, 557, 560, 561, 580, 581, 583, 584
  Benutzerverwaltung 543
  Bibliotheksverwalter 398
  Bildersammlung 386
  Datei 334
  Deklaration 281
  Einstellungen 442
  Fenster 584
  FUP/KOP/AWL 307
  Geräte 282, 287, 293
  Gerätekommunikation 295
  Hilfe 384
  Installation 387

Intelligente Code-Bearbeitung 495, 497
Lesezeichen 238
Meldungsfenster 405
Objekte 406, 413, 414
Online 417
Quelltextnavigation 246, 247
Rezepturverwalter 494
Rückgängig/Wiederherstellen 542
Suchen/Ersetzen 328
Übersetzen 248
Zwischenablage 279
Befehlskategorie 479
Befehlskategorie 482
Befehlskategorie Ansicht 583
Befehlskategorie Installation 392
Befehlskategorie Texteditor 526
Befehlskategorie Textlisten 528
Befehlskategorie Trace 536
Bei Quellcodeverwaltung anmelden 525
Beim Starten 443
Beispielapplikationen Softmotion 1001
Benutzer 179
Benutzer abmelden 545
Benutzer bearbeiten 378, 917
Benutzer hinzufügen 378, 917
Benutzer importieren 917
Benutzer löschen 917
Benutzer Owner 179, 378
Benutzer und Gruppen
Projekteinstellungen 372
Benutzer und Gruppen 378
Benutzer und Gruppen 917
Benutzerdefinierte Attribute 1057
benutzerdefinierter Datentyp 1051
Benutzereingabe 843
Benutzergruppe 917
Benutzergruppen 145
Benutzerkonfiguration speichern 917
Benutzerkonto 378, 917
Benutzerkonto wechseln 543
Benutzername
ENI Server 525
Benutzername 179
Benutzername 543
Benutzername 917
Benutzeroberfläche
Fenster-Layout zurücksetzen 586
Internationale Einstellungen 452
Landessprache 1024
Sprache 452, 1024
Benutzeroberfläche anpassen 166
Benutzerverwaltung
Befehle 543
Einstellungen 378
Projekteinstellungen 372
Rechte 546
Benutzerverwaltung 179
Benutzerverwaltung 543
Benutzerverwaltung für Gerät 917
Bereinigen
Applikation 252
Beschleunigung
CAM 588
Beschreibung
Projektinformationen 366
Beschreibung Trace-Achse 825
Beschriftung der Achse in Trace-Aufzeichnung 825
Bestätigter Transfer 561
Bestellnummer Gerät 924
Bestimmtes Profil 372
Bewegung von Visualisierungselementen 843
Bezeichner
Mehrfachverwendung 1025
Bezeichner 1025
Bezeichner 1028
Bezeichnung für Datenbankobjekt 520

Bezeichnung für Projekt in Quellcodeverwaltung 524
Bibliothek
Abhängigkeiten 387
Ablageort 387
Allgemeine Information 139
Busspezifische Bibliotheken 1180
CAA Bibliotheken 1181
CAA_AsyncManager.library 1181
CAA_CallBack.Library 1181
CAA_CanL2.library 1181
CAA_CanOpenMan.library 1181
CAA_CanOpenStack.library 1181
CAA_CiA405.lib 1180
CAA_CiA405.library 1181
CAA_DTUtil.library 1181
CAA_FILE.library 1181
CAA_Mathematics.library 1181
CAA_MemBlockMan.library 1181
CAA_Memory.library 1181
CAA_NetBaseServices.library 1181
CAA_RealTimeClock.library 1181
CAA_ResMan.library 1181
CAA_SegBufferMan.library 1181
CAA_SerialCom.library 1181
CAA_Settings.library 1181
CAA_Tick.library 1181
CAA_Timer.library 1181
CAA_Types.library 1181
CoDeSys V3.x System Libraries 1180
Container 139
Dokumentation 139, 748
Eigenschaften... 403
eindeutiger Zugriff 139
erstellen und installieren 349
extern 139, 250
Generelles 139
gerätespezifische Referenz 139
Hinzufügen 399
IIoDrvAcyclicServices.library 1180
IIoDrvBusControl.Library 1180
IIoDrvDPV1C1.library 1180
IIoDrvDPV1C2.library 1180
IIoDrvProfibus.Library 1180
intern 139
Konvertierung 454
laden 404
Mapping für Konvertierung 338
Namensraum 139, 403
Platzhalter 399
Projektinformationen 366
referenziert 139
Runtime Error Codes 1180
SM3_Basic.Library 1180
SM3_CNC.Bibliothek 1180
SSysComAsync.library 1180
Standard.Library 1180
SysCom.library 1180
SysCPUHandling 1180
SysDir.library 1180
SysDirAsync.library 1180
SysEvent.library 1180
SysFile.library 1180
SysFileAsync.library 1180
SysInt.library 1180
SysMem.library 1180
SysPci.library 1180
SysPort.library 1180
SysPortAsync.library 1180
SysProcess.library 1180
SysSem.library 1180
SysSemProcess.library 1181
SysShm.library 1181
SysSocket.library 1181
SysSocketAsync.library 1181
SysTask.library 1181
Systembibliotheken 1180

SysTime.library 1181
SysTimer.library 1181
SysTypes.library 1181
Util.library 1180
Version 139
Versionshandling 403
versteckt 139, 403
Verwalten 748
zielsystemspezifisch 139, 399
Bibliothek 139
Bibliothek 387
Bibliothek entfernen 748
Bibliothek erneut laden 404
Bibliothek für AS 783, 784
Bibliothek für Visualisierungselemente 392
Bibliothek hinzufügen 399, 748
Bibliothek installieren 387, 392
Bibliothek konvertieren 338
Bibliothek verschlüsseln 349
Bibliotheken
Optionen 454
Bibliotheken für Visualisierungen 191
Bibliotheken Installation 387
Bibliothek-Menü 139
Bibliotheksdateien 347
Bibliotheks-Datenbank 387
Bibliothekseigenschaften 748
Bibliotheks-Handling vereinfachen 463
Bibliothekskategorie 387, 399
Bibliotheksmenü 751
Bibliotheksname 748
Bibliotheks-Namensraum 748, 1160
Bibliotheksprojekt 347, 349, 366
Bibliotheksreferenz
Platzhalter 399
Bibliotheksreferenz 403
Bibliotheksreferenz konvertieren 338
Bibliotheksreferenzen konvertieren 454
Bibliotheks-Repository
automatische Installation 349
Verfügbarkeit 463
Bibliotheks-Repository 139
Bibliotheks-Repository 387
Bibliotheksversion 139, 365, 399, 748
Bibliotheksversion aktualisieren 338
Bibliotheksverwalter
Befehle 398
Eingänge/Ausgänge 748
Graphisch 748
Bibliotheksverwalter 139
Bibliotheksverwalter 398
Bibliotheksverwalter 748
Bibliotheksverwaltung 463
Bibliotheksverzeichnis 139
Biblliothek
Deinstallieren 387
Installation 387
Bild 552
Bild für Visualisierungshintergrund 552
Bildauswahl 104
Bilddatei in Visualisierung 839, 843
Bilddateien
Verzeichnis in Visualisierungs-Optionen 474
Bilddateien 372
Bilddateien 474
Bilddateien für Visualisierung 474
Bilder in Visualisierung 190, 386
Bilder ohne Pfad registrieren 474
Bildersammlung 104, 190, 386, 474
Bildersammlung Befehle 386
Bildersammlung erzeugen 386
binär 441
Bis zur zugehörigen Klammer selektieren 527
BITADR 1131, 1157
Bitadresse 1131, 1157
Bitmap für Bausteinelemente in FUP/KOP 461

Bitmap für Objekt 561
Bitmap für Visualisierungshintergrund 104, 552
Bitmap ID Variable 104
bitmap ID variable for visualization elements 843
Bitmap in Visualisierung 843
bitmapinfo 843
Bit-Operation 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1159
Bitzugriff 704
Bitzugriff auf Variable 1167
Blocktreiber 298, 901
BOOL 1088
BOOL_TO Konvertierungen 1133
BOOL_TO_DATE 1133
BOOL_TO_DT 1133
BOOL_TO_INT 1133
BOOL_TO_STRING 1133
BOOL_TO_TIME 1133
BOOL-Konstanten 1162
Bootapplikation 134
Bootapplikation Einstellungen 561
Bootapplikation erzeugen 422
Bootprojekt 134, 422
Bootprojekt löschen 441
Branchement
  Alternative 503
  Parallèle 503
breakpoint 794
Breakpoint
  CFC Editor 623
  FUP/KOP/AWL Editor 742
Breakpoint 137
Breakpoint 742
Breakpoints Befehle 240
Breite des Operandenfeldes 461
Breite von AS-Aktionen 467
Breite von AS-Attributen 467
Breite von AS-Qualifizierern 467
Breite von AS-Schritten 467
Breite Zelle in Seiten-Layout 485
Broadcast Adresse 120
Broadcast für EtherCAT 951
Broadcast-Adresse 561
Broadcasting 125
browse location 561
Busspezifische Bibliotheken 1180
Busteilnehmer 987
Buszyklus-Optionen 909
Buszyklus-Task 909
button 843
button height 843
buttonheight 843
buttonstate variable 843
BYTE 1088
BYTE_TO_BOOL 1135
Byte-Adressierung 1169
C Anweisung 713
c code 250
C in CoDeSys 57
C Modifikator 713
c routine 250
C Stub-Date 250
CAA Bibliotheken 1181
CAA_AsyncManager.library 1181
CAA_CallBack.Library 1181
CAA_CanL2.library 1181
CAA_CanOpenMan.library 1181
CAA_CanOpenStack.library 1181
CAA_CiA405.lib 1180
CAA_CiA405.library 1181
CAA_DTUtil.library 1181
CAA_FILE.library 1181
CAA_Mathematics.library 1181
CAA_MemBlockMan.library 1181
CAA_Memory.library 1181
CAA_NetBaseServices.library 1181

CAA_RealTimeClock.library 1181
CAA_ResMan.library 1181
CAA_SegBufferMan.library 1181
CAA_SerialCom.library 1181
CAA_Settings.library 1181
CAA_Tick.library 1181
CAA_Timer.library 1181
CAA_Types.library 1181
CAL 310, 1132
CAL Operator in AWL 713
Call Stack 559
CAM
  Anlegen 588
  Beschleunigung 588
  Datenstrukturen 607
  Definition 588
  Durchlauf 588
  Eigenschaften 588
  Elementeigenschaften 588, 611
  generisch 608
  Geschwindigkeit 588
  Jerk 588
  Kompilieren 588
  Kompilierungsformat 561
  manuell erzeugt 610
  Master-Position 609
  Nocke 588
  Position 588
  Punkt 588
  Slave-Beschleunigung 609
  Slave-Geschwindigkeit 609
  Slave-Position 609
  SMC_CAMXYVA 609
  Struktur MC_CAM_REF 608
  über Applikation erzeugt 610
  Werkzeugbox 588
CAM 253
CAM 588
CAM 611
CAM Befehle 253
CAM Editor 253, 588, 598, 607, 608, 609, 610, 611
CAM Eigenschaften 561
CAM Online-Datei lesen 254
CAM Online-Datei schreiben 253
CAM-Datein in ASCII-Tabelle schreiben 253
CAM-Daten aus ASCII-Tabelle lesen 254
CAM-Editor 989
CAN Gerät 925
CAN Knoten 925
CAN Konfiguration
  CANopen Gerät 936
  CANopen Manager 934
  Information 930, 935, 946, 947
  modulares Gerät 935
  nicht-modulares Gerät 935
  Status 930, 935, 946, 947
CAN Modul 925
CANbus
  Baudrate 924, 930
  CANopen I/O Mapping 946
  CANopen Modul 947
  Netzwerknummer 924, 930
  Parameter 930
CAN-Bus 925
CANbus Knoten Konfiguration
  CANbus 924, 930
  CANbus Konfiguration 930
CANbus Konfiguration 930
CAN-Bus Konfigurationseditor 925
CANopen 52
CANopen Gerät
  CANopen Konfiguration 945
  CANopen Remote Device 936
  Emergency 936
  Empfangs PDO Mapping 939
  Heartbeat 936

Initialisierung 936
optional 936
Parameter 945
Sende PDO Mapping 943
Service Data Object 943
CANopen Gerät 935
CANopen I/O Abbild 935
CANopen I/O Mapping 946
CANopen Konfiguration
Gerät einfügen 925
Hierarchie 925
CANopen Konfiguration 925
CANopen Konfiguration 945
CANopen Manager
CANopen I/O Abbild 935
Common.CANbus Konfiguration 934
CANopen Manager 925
CANopen Manager 934
CANopen Modul
I/O Abbild 947
CANopen Modul 946
CANopen Modul Konfiguration 946
CANopen Remote Device 936
CASE 803
CASE-Anweisung 799
center 843
CFC
Anschlüsse zurücksetzen 267
Arbeitsblatt 621
Arbeitsblatt bearbeiten 256
Gittereinheiten 256
Größen 256
offset 256
Optionen 459
VAR_IN_OUT 267
VAR_INPUT CONSTANT 266
CFC Ausführungsreihenfolge 255
CFC Editor
Ausführungsreihenfolge 621
Ausgang 616
Baustein 616
Breakpoint 623
Cursor-Positionen 614
Eingang 616
Element auswählen 621
Element einfügen 621
Element kopieren 621
Element löschen 621
Element verschieben 621
Elemente anordnen 621
Element-Text editieren 621
Haltepunkt-Positionen 623
Kommentar 616
Komposition 616
Marke 616
Monitoring 623
Online-Modus 623
RETURN 616
Selektor 616
Sprung 616
CFC Editor 614
CFC Editor Befehle 255
CFC Editor Optionen 459
CFC Elemente 616
CFC Menü 255
CFC Werkzeuge 616
cfg-Datei 108
charset 843
check functions 107
CheckBounds 71
CheckDivByte 71
CheckDivDWord 71
CheckDivReal 71, 1113
CheckDivWord 71
CheckPointer 1093, 1107
CheckRangeSigned 71, 1103

CheckRangeUnsigned 71, 1103
Client-Breite 895, 896
Client-Höhe 895, 896
clipping 843
CMP ID 908
CNC
  Bedienfeld 677
  DIN66025 635
  Eigenschaften Bahnelement 679
  Eigenschaften des CNC-Objekts 684
  Grafischer Editor 675
  Online 690
  Parameter-Editor 693
  Pathview-Editor 675
  Tabellendarstellung 673
  Tabelleneditor 673
  Texteditor 633
  Werkzeuge 679
CNC Info 687
CNC Online 690
CNC_file.project 1023
CNC_table.project 1022
CNC-Befehe
  Programm neu nummerieren 268
CNC-Befehel
  Bahnansicht scrollen 268
CNC-Befehle
  Dynamik analysieren 268
  Endpunkte anzeigen 268
  Interpolationspunkte anzeigen 268
  Programm aus ASCII-Datei laden 268
  Programm drehen 268
  Programm in ASCII-Datei schreiben 268
  Programm skalieren 268
  Programm verschieben 268
  Raster anzeigen 268
  Richtung umdrehen 268
  Satzunterdrückung 268
  Variaben setzen 268
  Von DXF-Datei importieren 268
  Vorverarbeitete Bahn anzeigen 268
CNC-Befehle 268
CNC-Editor Werkzeuge 679
CNC-Programm neu nummerieren 274
CNC-Sprache 635
COB-ID 936, 939
Code für <applikation> erzeugen 249
Codeerzeugung für AS 561
Codegenerierung 134
Code-Generierung 1078
Code-Generierung für AS 470
Code-Generierung offline 249
Code-Komplettierung 445
CoDeSys 57
CoDeSys 2.3 Konverter 457
CoDeSys HMI 194, 196
CoDeSys Konzepte 57
CoDeSys SP Dienst 147
CoDeSys SP service 147
CoDeSys V2.3 Projekte 59, 63
CoDeSys V3.x System Libraries 1180
codesys.exe starten 1024
coil/set and reset 741
color variables 843
colors 843
COM Port für Gateway 901
Commandes d'éditeur SFC 500
Common.CANbus Konfiguration 934
communication cycle in CANbus 936
compiled library files 349
compiled-library Dateien 349
Compile-Optionen 372
Compiler-Definitionen 561, 1078
Compiler-Version 338, 365, 372
CONSTANT 497, 1045, 1048
Container-Bibliotheken 139

CONTINUE 804, 805, 806, 807
Continuous Function Chart 614
COS 1145
Cosinus 1145, 1146
C-Rahmendatei für externe Bibliothek 250
CreateRecipe 758
CreateRecipeNoSave 758
C-Routinen 57
C-Schnittstellen 139
culture 1024
Cursor
  Trace 537
Cursor 537
Cursor-Positionen in AS 767
Cursor-Positionen in CFC 614
Cursor-Positionen in FUP 727
Cursor-Positionen in KOP 727
curve 843
D 783, 1090, 1169
d# 1163
dar-Datei 921
Darstellung im Online-Modus 441
Darstellung Target-Visualisierung 895
Darstellungsart für Trace-Kurve 825
Darstellungsmodus
  Pragma 1059
Darüber einfügen 321
Darunter einfügen 320
Data control time 978
Data Server
  Hinzufügen 695
Data Server 125, 694
Data Server 697
Data Source
  Datenquellenvariablen 697
  Kommunikation 697
Data Source 190
Data Source 697
Data Type Unit (DUT) 82
DataSource Editor plug-in 694
DataSource Object plug-in 694
DATE 1090
DATE Konstanten 1090
DATE_AND_TIME 1090
DATE_AND_TIME Konstanten 1090
DATE_AND_TIME Konvertierungen 1139
DATE_AND_TIME-Konstanten 1164
DATE_TO Konvertierungen 1139
Datei
  Neu 335
  Ort 335
  Projekteinstellungen 372
  Projektinformationen 366
  Sicherungskopie 338
  Transfer 906
Datei als Text einfügen 526
Datei Befehle 334
Datei für Rezeptur 754
Datei Menü 171
Dateieigenschaften 96
Dateien für Textlisten 474
Dateien für Textlisten ohne Pfad registrieren 474
Dateisystem-Objekte 921
Datei-Tansfer zu/von Steuerung 906
Dateityp
  Projekt 338
DATE-Konstanten 1163
Datenaustausch 125
Datenbank
  Einchecken 518
  Verknüpfung 234
Datenbank 234
Datenbank 473
Datenbank für Geräte 293
Datenbankaktionen 473, 522
Datenbankanbindung eines Objekts 561

Datenbankobjekt Versionsgeschichte 520
Datenbankprojekt
  Bezeichnung 524
Datenbankverbindung 514
Datenbankverbindung ändern 515
Datenfluss in CFC 261
Datenquelle
  Hinzufügen 696
  Konfiguration 697
  Typ 696
  Visualisierung 704
Datenquelle 125, 694
Datenquellenvariablen 697
Datenstrukturen CAM 607
Datentyp
  benutzerdefiniert 82, 1051
  erweitert 82
  Obsolete-Pragma 1072
  Querverweisliste 555
  Warnungsmeldung 1072
Datentyp 708
Datentyp 1048
Datentypen
  ANY_TYPE 1091
  Array 1097
  Aufzählung 1101
  Enumeration 1101
  Feld 1097
  LTIME 1091
  normerweiternd 1090
  Pointer 1093, 1107
  Referenz 1092, 1106
  Struktur 1100
  UNION 1091
  WSTRING 1091
  Zeiger 1093, 1107
Datentyp-Name 1028
Datentyp-Objekt 82
DC beim EtherCAT Slave 955
DC Zykluseinheit-Kontrolle 955
DCF 936
DCF-Datei 936
Deaktivieren
  Haltepunkt 557
Debug Menü 417
debugging 240, 243
Debugging
  Aufrufliste 559
  CFC Editor 623
  FUP/KOP/AWL Editor 742
  ST-Editor 794
Debugging 137
Deckblatt 484
Deckblatt bearbeiten 485
Deckblatt für Projektdokumentation 485
default settings for SFC 561
default slave ratio for heartbeat time in percent 934
define
  pragma 1078
define 1078
defined 1078
Dehnen 539
Deinstallation 146
Deinstallation eines Geräts 293
Deinstallieren
  Visualisierungselemente 392
Deklaration
  Autodeklaration 452, 1035
  Dialog 497
  Funktionen 1051
  Kurzform-Modus 1035
  Methoden 1051
Deklaration 1025
Deklaration einfügen 281
Deklaration mit Adresszuweisung 1037
Deklaration nach unten verschieben 281

Deklarationsdialog 497
Deklarationseditor 708, 1035
Deklarationseditor Ansicht speichern 460
Deklarationseditor Optionen 460
Deklarationskopf editieren 281
Deklarations-Position finden 247
Deklration nach oben verschieben 281
DeleteRecipe 758
DeleteRecipeFile 758
Den Ursprung relativ verschieben 256
Dereferenzierung 1093, 1107, 1132
Details
  Applikation 134
Details der Projektversionsgeschichte 522
Details für Datenbankobjekt 520
Details zu Bibliothek 387
devdesc.xml Datei 108, 925
devdescr.xml Datei 293
Device Description Datei 108, 293
device description file 925
Device Repository Plug-In 293
dezimal 441
Diagnose für Gerät 924
Diagnose Nachricht 951
Diagnoseinformation
  Profibus-Gerät 988
Diagnose-Visualisierung
  Einschränkungen 194
Diagnose-Visualisierung 181
Diagnose-Visualisierung 194
Diagnose-Visualisierung 704
Dialog für Haltepunkte 557
Dialog öffnen 843
Dialog schließen 843
Dialoge in Visualisierung 227
DialogManager 227
Dialog-Visualisierung 227
Die Zugangsdaten speichern 525
Dieselbe Sprache wie Microsoft Windows verwenden 452
DIN66025
  Sätze 635
  Satznummer 635
DIN66025 635
DINT 1088
displaymode 1059
DIV 1113
Division 1113, 1115
Do not get the latest version 516
Dokumentation
  Datenquelle 697
Dokumentation anzeigen
  Datenquelle 697
Dokumentation anzeigen 495
Dokumentation für Bibliotheksbaustein 748
Dokumentation von Bibliotheksbausteinen 139
Dokumentieren 484, 485
Dokumentieren eines Projekts 237
download
  remanente Variablen 1045
download 1053
Download 424, 427
DP-Master
  DP-Parameter 978
  Gruppeneigenschaften 978
DP-Master 986
DP-Modul 985
DP-Modul Konfiguration 987
DP-Parameter
  DP-Modul 985
  DP-Slave 982
DP-Parameter 978
DP-Slave
  DP-Parameter 982
DP-Slave 986
Drehpunkt eines Visualisierungselements 843

Drehung 485
Drehung von Visualisierungselementen 843
Drive Interface 992
Drive Logical 997
Druckbereich 493
Drucken 139, 237, 485, 493
Drucken aus Visualisierung 843
Drucken Befehle 483
Druckversion 484
DS 783
DT 1090
dt# 1164
DT_TO Konvertierungen 1139
dum-Datei 917
durchschnittliche Zykluszeit 818
DUT 82
DWORD 1088
DXF-Datei 275
dynamic texts 843
dynamic variables 843
Dynamik analysieren 276
dynamische Abbildung 104
dynamische Erzeugung von Instanzen 1148, 1150
dynamische Texte 843
Dynamische Texte 187
dynamischer Test 98
E/A Abbild EtherCAT Master 954
E/A Abbild EtherCAT Slave 965
E/A aktualiseren bei Gerätestop 909
E/A-Abbild 911
E/As aktualisieren im Stop 909
E/A-Werte 909
Ebene entfernen 485
Ebene hinzu 485
Edit Commands Plug-In 279
Edit Control in Visualisierung 220
Editor
  nächster 585
  vorheriger 585
  Werkzeuge 583
Editor für Kurvenscheiben 588
Editor öffnen 582
Editor schließen 586
Editorfenster horizontal 587
Editorfenster vertikal 587
Editorfenster wechseln 585
Editorteilfenster wechseln 586
eds-Datei 108, 293
EDS-file 936
Eigenschaft
  Methode 86
  Monitoring 1067
Eigenschaft 86
Eigenschaft 1067
Eigenschaft anzeigen im AS-Editor 467
Eigenschaft eines AS-Elements 581
Eigenschaften
  Allgemein 561
  AS-Einstellungen 561
  Bibliothek 403
  Bitmap 561
  Bootapplikation 561
  CAM 561, 588
  Externe Datei 96
  Funktionen 366
  Haltepunkt 557
  Kurvenscheibe 561
  Netzwerkeinstellungen 561
  Netzwerkvariablen 561
  Objekt 64, 561
  POU 561
  Projektinformationen 366
  Quellcodeverwaltung 561
  Schlüssel 366
  Übersetzen 561
  Visualisierung 561

Visualisierungselement 843
Visualisierungselemente 583
Zugriffskontrolle 561
Eigenschaften 561
Eigenschaften AS-Element 771
Eigenschaften Bahnelement 679
Eigenschaften Benutzerkonto 378
Eigenschaften des CNC-Objekts 684
Eigenschaften einer Bibliothek 748
Eigenschaften eines CAM-Elements 611
Eigenschaften eines Visualisierungselements 583
Eigenschaften Heartbeat Consumer ändern 936
Eigenschaften Taskkonfiguration 813
Eigenschaftenbreite 467
Eigenschaften-Dialog Visualisierung 843
EinAusgabevariablen 1043
Einblenden 585
Einchecken 518, 582
Einchecken automatisdch 473
Eine Datei pro Teilbaum 411
Eine Seite pro Objekt 485
EINF 163
einfache Linie 843
Einfang in CFC 616
Einfügemodus 526
Einfügen
Visualisierungselement 839
Einfügen 280, 281
Einfügen in AS 768
Einfügen in AWL 720
Einfügen von Elementen in FUP/KOP/AWL 730
Einfügeposition auswählen 108
Einfügeposition in FUP/KOP/AWL 731
Eingabeadresse 1169
Eingabedialog in Visualisierung 843
Eingabehilfe
verstecken 1062, 1063
Eingabehilfe 495
Eingabekonfiguration 843
Eingabekonfiguration für Visualisierungselemente 843
Eingaben auf Visualisierungselement 843
Eingabeunterstützung im Texteditor 1087
Eingabevariablen 1042
Eingabevariablen in Visualisierung 186
Eingang in FBD/LD 316
Eingänge EtherCAT Master 954
Eingänge EtherCAT Slave 959, 965
Eingänge/Ausgänge Bibliotheksbaustein 748
Eingangsaktion 771, 772
Eingangsdaten Adresse 951
Eingangskanal 911
Eingangslänge 985
Eingangsspeicher 1169
Eingangsvariablen 1042
Einloggen 419
Einloggen auf Applikation 419
Einloggen in Quellcodeverwaltung 525
Einrückung 445
Eins vor 261
Eins zurück 261
Einstellungen Benutzeroberfläche 442
Einstellungen für Benutzerkonten 378
Einstellungen für Intelligentes Kodieren 452
Einstellungen für Projekt 372
Einstellungen Texteditor 445
Einzelschritt 244
Einzelschrittabarbeitung 137
Einzelzyklus 427
Einzug 445
Einzugsgröße 445
Element
Visualisierung 839
Visualisierungselement selektieren 841
Element auswählen in CFC 621
Element einfügen in CFC 621

Element in Visualisierung 843
Element installieren 392
Element löschen in CFC 621
Element verschieben in CFC 621
Elemente anordnen in CFC 621
Elemente einfügen in FUP/KOP/AWL 730
Elemente für FUP/KOP/AWL 731
Elemente in AS 772
Elemente mit geraden Linien verbinden 461
Elementeigenschaften
  CAM 588, 611
  Datenquellenvariablen 704
Elementeigenschaften 581, 583
Elementeigenschaften in AS 771
Elementeigenschaften Visualisierungselement 843
Elementhintergrund 454
Elementliste 552, 890
elementlook 843
Elementname 890
Elementname in Visualisierung 843
Elementnummer in CFC 261, 264, 265
Elementnummern in CFC 621
Elementschriftstil 454
Elementtyp 890
Elementtypen in Visualisierung 843
Elementvordergund 454
ellipse 839, 843
Ellipse 653
Emergency-Nachrichten 936
Empfangs PDO Mapping 939
Empfindlichkeit
  Task-Watchdog 814
EN/ENO 258, 310
EN/ENO in CFC 258
enable cycle 936
enable-Ausgang in CFC 258
enable-Eingang in CFC 258
END_STRUCT 1100
END_UNION 1091
Endpunkte anzeigen 270
Engineering Interface 234
ENI 234, 509, 510
ENI Client 234
ENI Server 234, 473
ENI Server Anmeldung 525
Enumeration 1101
Enumerations-Namensraum 1161
EQ 1129
EQ Operator in AWL 713
Ereignis
  Task 814
Ereignis 814
Ereignisgesteuerte Übertragung 561
Ereignistask 814
Ereigniszeit 939
erlauben 921
Erlaubt 546
Ersetzen 328, 332
Ersetzungsdatei importieren 530
Erstellen Befehle 248
Erstellen Menü 171, 248
erweiterbare Funktion 71
erweiterbare Funktionen 1051
Erweitert 73, 77, 82
Erweiterte Aktualisierung 392
Erweiterte Einstellungen
  Quellcodeverwaltung 510
Erweiterter Strukturierter Text 797
Erweiterung 73
Erweiterung eines Funktionsblocks 77, 79, 80
Erzeuge Laufzeitsystemdateien 250
ETC_CO_SdoRead 970
ETC_CO_SdoRead_Access 973
ETC_CO_SdoRead4 971
ETC_CO_SdoReadChannel 973
ETC_CO_SdoReadDWord 973

ETC_CO_SdoWrite 973
ETC_CO_SdoWrite4 975
ETC_CO_SdoWriteAccess 976
ETC_CO_SdoWriteDWord 976
ETCSlave 967
EtherCAT
  Bus anhalten 966
  IODrvEtherCAT.library 969
  Mailbox Nachricht 976
  Master-Instanz 966
  Parameter lesen 970, 971, 973
  Parameter schreiben 973, 975, 976
  read and write parameters 969
  Slave-Instanz 967
EtherCAT 52
EtherCAT Address 955
EtherCAT Adresse 955
EtherCAT I/O Abbild 965
EtherCAT I/O Mapping 965
EtherCAT Konfiguration
  Autokonfiguration 951
  Expertenmodus 951
EtherCAT Konfiguration 953
EtherCAT Konfiguration 965
EtherCAT Konfigurationsstatus 966
EtherCAT Master
  LRW anstelle von LWR/LRD verwenden 951
  Meldungen pro Task aktivieren 951
  Neustart 966
  Optionen 951
  Slaves automatisch neu starten 951
EtherCAT Master I/O Abbild 954
EtherCAT Master Information 954, 965
EtherCAT Master Konfiguration 951
EtherCAT Master Parameter 953
EtherCAT Slave
  Autokonfiguration 955
  DC Zykluseinheit-Kontrolle 955
  expert settings 958
  Experteneinstellungen 955, 958, 959
  FNNU/Sync 958
  Startüberprüfung 955
  Stations-Alias 955
  Status 967
  Timeouts 955
  Verteilte Uhren 955
  Zeitüberwachung 955
EtherCAT Slave 955
EtherCAT Slave 958
EtherCAT Slave 959
EtherCAT Slave Parameter 965
EtherCAT XML erzeugen 285
Event in Visualisierung bemerken 220
Event-Logging 908
Everyone 378
EXIT 807
exit Methode 84
Exit Methode 1069
EXIT-Anweisung 799
Exit-Methode 1053
EXP 1144
ExpandFully 1060
Experteneinstellungen 958
Expertenmodus EtherCAT Konfiguration 951
Exponentialfunktion 1144
Export Log-Datei 908
Export Textliste 530
Export vor Übersetzen 561
Exportdatei 411, 412
Exportdatei für Textliste 530
Exportiere PLCOpenXML 414
Exportieren 411
Exportieren von Textlisten 98
expr 1078
EXPT 1147
ExST 797

EXTENDS 77, 82
extern 92
externe Bibliothek 139, 250
Externe Datei
  Eigenschaften 96
  Hinzufügen 96
Externe Datei 96
externe Datenbank 234
Externe Datenquellen 190
Externe Funktion 92
Externe Implementierung 561
Externe Methode 92
externe Variablen 1044
Externer Funktionsblock 92
Externes Ereignis für Tasks 814
Extrahiere Archiv 352
F1 146
Fahrbefehl 640
fallende Flanke 322
FALSE 1162
Farbdefinition 843
Farbe für Trace-Kurve 825
Farben für Trace-Aufzeichnung 825
Farben im Texteditor 445
Farbvariablen 843
fb_exit 1069
FB_exit Methode 84, 1053
FB_init Methode 84, 1051
FB_reinit Methode 84, 1051
FBD/LD/IL
  Baustein 313
Features 463
Feature-Set 57
Feature-Sets 463
Fehler in AS 784
Fehleranalyse in AS 784
Fehlermeldung 580, 1055
Fehlersuche 137
Fehlerverhalten SDO-Transfer 943
Felder 1097
Fenster
  Alle Editoren schließen 586
  Editor schließen 586
  Nächster Editor 585
  Nächstes Teilfenster 586
  Neue horizontale Registerkartengruppe 587
  Neue vertikale Registerkartengruppe 587
  Unverankert 585
  Vorheriger Editor 585
  Vorheriges Teilfenster 586
Fenster 587
Fenster <n> 587
Fenster andocken 585
Fenster ausblenden 166, 585
Fenster Befehle 584
Fenster einblenden 585
Fenster im Online-Modus 169
Fenster Menü 171
Fenster öffnen 166
Fenster positionieren 166
Fenster schließen 166
Fenster-Dialog 587
Fenster-Layout 586
Fenster-Layout zurücksetzen 586
Fenster-Menü 166
Feste Größe für Operandenfeld 461
file close 345
file merge 354
file new 335
file save 346
file saveas 347
Files-Transfer 906
fill color 843
fillattributes 843
Filter für Projektgeschichte 520
Firma 366, 399

Flag-Optionen für AS 561
Flags
AS 470
Flags in AS 784
Flankenerkennung 322
FMMU 958
Folgende Größen verschieben 256
Font 187
font flags 843
Font für Visualisierung 843
font height 843
font name 843
font variables 843
Font-Attribute 843
Fontvariablen 843
FOR 804
FOR-Anweisung 799
Forceliste für Applikation aufheben 436
Forcen 432, 435, 436, 794
Forcen aufheben 432, 435, 436
Format
Bibliotheksreferenzen konvertieren 454
Konvertieren alter Projekte 338
Projektdatei 338
Format 454
Format Papier 485
Formatierung von Texten 98
Formatierung Zelle in Seiten-Layout 485
Formatierungsangaben für Visualisierungstexte 187
FOR-Schleife 804
Frame 186, 843, 884
Frame Auswahl 550, 839
frame color 843
Frame Konfiguration 550
Frame Parameter aktualisieren 884
frame reference 843
frameattributes 843
Frame-Konfiguration 843
Framevisualisierung umschalten 843
Frame-Visualisierung umschalten 550
Freeze-Modus 978
freilaufend 814
freilaufende Task 814
Füllung Fußzeile 485
Füllung Kopfzeile 485
FUNCTION 71
function block diagram 326
function call 71
FUNCTION_BLOCK 73, 77
Funktion
Adressoperator ADR 1093, 1107
extern 92
Hinzufügen 71
PARAMS OF <data type> 71
zusätzliche Ausgäng 71
zusätzliche Ausgänge 1042
Funktion 71
Funktion als Operand 1172
Funktion in FUP/KOP/AWL 735
Funktionalitätsumfang 57
Funktionsaufruf in AWL 720
Funktionsbaustein
Eigenschaft 84
Erweiterung 77
extern 92
hinzufügen 73
Initialisierung 1064
keine virtuellen Aktionen 1070
Methode 84
Funktionsbaustein 73
Funktionsbaustein in FUP/KOP/AWL
Tooltip zu Parametern 716
Funktionsbaustein in FUP/KOP/AWL 326
Funktionsbaustein in FUP/KOP/AWL 735
Funktionsblock
Eigenschaft 84, 86

Erweiterung 77
Haltepunkte 241
hinzufügen 73
Initialisierung 1051
Methode vor Download 1053
Methoden 84
Methodenaufruf 80
Schnittstellen 79
Funktionsblock 73
Funktionsblock-Aufruf 75
Funktionsblock-Aufruf in ST 801
Funktionsblock-Instanz 74, 75
Funktionsblock-Instanz kopieren 1053
Funktionsblock-Name 1028
Funktions-Name 1028
Funktionsplan 711
Funktionspointer 1093, 1107
Funktionszeiger 1093, 1107, 1131, 1157
FUP 711
FUP Befehle 730
FUP Editor Optionen 461
FUP/KOP/AWL
Baustein 735
Leitungsverzweigung 737
Marke 735
Nachverschaltung 737
Netzwerk 733
Nicht verwendete FB-Aufrufparameter entfernen 326
RETURN 736
Sprung 735
Zuweisung 734
FUP/KOP/AWL Editor
Breakpoint-Positionen 742
Einfügen 320, 321
Haltepunkt-Positionen 742
monitoring 742
negierten Kontakt parallel einfügen unterhalb 319
negierter Kontakt 319
Online-Modus 742
Reset-Spule einfügen 317
Set-Spule einfügen 317
Sprungmarke einfügen 316
Spule einfügen 317
FUP/KOP/AWL Editor Befehle 307
FUP/KOP/AWL Einfügepositionen 731
FUP/KOP/AWL Menü 730
FUP/KOP/AWL Toolbox 731
FUP/KOP/AWL Werkzeuge 731
FUP/KOP/AWL-Editor
Tooltip 716
FUP/KOP/AWL-Editor umschalten 326
FUP-Ansicht 326
FUP-Editor
Cursor-Positionen 727
FUP-Editor 716
Fußzeile 485
g_CNCProgManager 688
g_CNCQueueManager 688
G1 642
G2 644
G20 657
G3 644
G36 659
G37 659
G4 647
G40 661
G41 661
G42 661
G50 661
G51 661
G52 661
G53 664
G54 664
G55 664
G56 664

G6 651
G60 661
G61 661
G75 666
G90 666
G91 666
G98 666
G99 666
Ganzer Bildschirm 579
ganzzahlige Datentypen 1088
ganzzahlige Typen konvertieren 1136
Gap 978
Gateway
  Bearbeiten 305
  Blocktreiber 298
  Hinzufügen 297
  Konfiguration 298
  zum lokalen Gateway verbinden 298
Gateway Adresse 901
Gateway bearbeiten 305
Gateway Dienst 147
Gateway hinzufügen 297, 901
Gateway konfigurieren 901
Gateway Router 298
Gateway Service 147
Gateway Treiber 901
Gateway verbinden 901
gateway.cfg 901
Gateway-Einstellungen 901
G-Code
  3D-Modus 655
  Ändern von Variablenwerten 659
  Ellipse 653
  Fahrbefehl 640
  G1 642
  G10 648, 655
  G15 655
  G16 655
  G17 655
  G18 655
  G19 655
  G2 644
  G20 657
  G3 644
  G36 659
  G37 659
  G4 647
  G40 661
  G41 661
  G42 661
  G5 648
  G50 661
  G51 661
  G52 661
  G53 664
  G54 664
  G55 664
  G56 664
  G6 651
  G60 661
  G61 661
  G75 666
  G8 653
  G9 653
  G90 666
  G91 666
  G98 666
  G99 666
  Gerade 642
  H 668
  H-Funktion 668
  K 657
  Koordinatenverschiebung 664
  L 657
  M 670
  M-Funktion 670

Modi 666
Parabel 651
Positionierung 640
R 672
Schaltpunkt 668
Spline 648
Sprünge 657
Verwendung von Variablen 672
Verzugszeit 647
Vorverarbeitung 661
Zeitsynchronisation mit Interpolator 666
G-Code 635
GE 1129
GE Operator in AWL 713
Gehe zu Definition 246, 247
Gehe zu Quelltextposition 405
Gehe zu Zeile 526
Gehe zur Definition 555
Gehe zur Quellcodeposition 557
Gehe zur zugehörigen Klammer 527
Generische Konfigurationsdialoge anzeigen 987
Generische Konfigurationseditoren anzeigen 466, 986
Gerade 642
Gerät
Adresse auflösen 901
ausgewähltes aus Liste löschen 305
Basisadresse 911
Benutzerverwaltung 917
Beschreibungsdatei 108, 293
Bestellnummer 924
Darstellung im Gerätebaum 108
Deinstallation 293
Device Description Datei 108
Hersteller 283
Information 924
Installation 108, 293
Instanz 911
Logging 908
Mapping für Konvertierung 338
Name 283
Repository 293
Simulation 436
Speicherort 293
urspüngliches durch neues ersetzen 457
Verbindung herstellen 441
Verbindung trennen 441
Version 283
Zugriffsrechte 921
Gerät 108
Gerät 282
Gerät 287
Gerät 293
Gerät aktualisieren 283, 285
Gerät anhängen 108, 283
Gerät auf Ursprung zurücksetzen 441
Gerät deaktivieren 286
Gerät einfügen 108, 283, 284
Gerät einstecken 283
Gerät für CANopen
CANopen Remote Device 936
modular 935
nicht-modular 935
Gerät hinzufügen 284, 285, 901
Gerät installieren 293, 463
Gerät optional 936
Gerät setzen 283, 285
Geräte Ansicht 560
Geräte Befehle 282, 287, 293
Geräte identifizieren 287
Geräte Konfiguration 993
Geräte suchen 287, 901
Geräte taufen 287
Gerätebaum
Regeln 108
Gerätebaum 57
Gerätebaum 108

Gerätebaum 282
Gerätebaum 287
Geräte-Benutzerverwaltung 917
Gerätebeschreibung 108
Gerätebeschreibungsdatei 293
Geräte-Datenbank 293
Gerätediagnose 924
Geräte-Dialog
  Applikationen 905
  Benutzer und Gruppen 917
  Kommunikationseinstellungen 901
  Zugriffsrechte 921
Geräte-Editor
  CAN-Bus Konfiguration 925
  Dateitransfer 906
  Files 906
  Kommunikationseinstellungen 901
  Konfiguration 904
  Status 924
Geräte-Editor 108
Geräte--Editor
  Log 908
Geräteeditor Optionen 466
Geräteeditor SoftMotion 993
Geräteeinstellungen 909
Geräte-Fenster 163, 560, 561
Geräte-Kategorie 924
Gerätekommunikation
  Aktiven Pfad setzen 306
  ausgewähltes Gerät löschen 305
  Gateway 298
  Gateway bearbeiten 305
  Gateway hinzufügen 297
  Gerätenamen ändern 298
  Netzwerk durchsuchen 306
  Zum lokalen Gateway verbinden 298
Gerätekommunikation 295
Gerätekommunikation Befehle 295
Gerätename
  Ändern 298
Gerätename 287
Geräte-Name 924
Gerätenamen ändern 298
Geräteobjekte 57
Geräteparameter 904
Gerätereferenz konvertieren 338, 457
Geräte-Repository
  Verfügbarkeit 463
Geräte-Repository 108, 293
Geräte-Speicherort 293
gerätespezifische Bibliothek 139
Gerätetyp 287
Geräte-Typ 924
Geräte-Version 924
Geräteverwaltung 463
Geschwindigkeit
  CAM 588
Gestion des utilisateurs 179
getbooleanproperty 366
getcompany 366
get-Eigenschaft 86
GetLiveList 987
getnumberproperty 366
GetRecipeCount 758
GetRecipeNames 758
GetRecipeValueNames 758
GetRecipeValues 758
gettextproperty 366
gettitle 366
getversion 366
getversionproperty 366
getypte Konstante 1048
getypte Konstanten 1166
Gierwinkel der Kamera ändern 677
Gitter im CFC 256
Gitter in Trace-Aufzeichnung 825

Gleichheit 1129
Gleitpunktpräzision 445
Gliederung 445
Global StartAdr 958
Global_init_slot 1061
Globale Bildersammlung 104
Globale Bildersammlung erzeugen 386
Globale Netzwerkvariablenliste 93, 125
Globale Textliste anlegen 528
globale Variablen
  Initialisierung 1043
globale Variablen 1043
globale Variablen 1044
globale Variablen in integrierter Visualisierung 194
globale Variablenliste
  Initialisierung 93
globale Variablenliste 93
globale Variablenliste 125
Globale Variablenliste Namensraum 1160
Globaler Namensraum-Operator 1159
GlobalImagePool 104, 386
GlobalTexList 528
GlobalTextList 98, 187
GNVL 93, 125
Gobale Variablenliste
  Qualified_Only 1075
Grafischer Editor 675
Groß-/Kleinschreibung 329
Großbuchstaben 527
Größe automatisch anpassen 256
Größe einer Variablen 1116, 1158
Größe von Visualisierungselementen 842, 843
größer als 1128
größer als oder gleich 1129
Gruppe 378, 549, 917
Gruppe bearbeiten 378, 917
Gruppe Everyone 179, 378
Gruppe hinzufügen 378, 917
Gruppe löschen 917
Gruppe Owner 179, 378
Gruppen
  Projekteinstellungen 372
Gruppen beim Profibus 978
Gruppen importieren 917
Gruppeneigenschaften 378
Gruppeneigenschaften DP-Master 978
Gruppieren nach Firma 399
Gruppieren nach Kategorie 387, 399
Gruppierung
  Visualisierungselemente 842
Gruppierung 549
Gruppierung aufheben 549
GS?-Datei 978, 982, 985
gsd-Datei 108, 293
GT 1128
GT Operator in AWL 713
Guard COB-ID 936
Guard Time 936
Gültigkeitsbereich 1025
Gültigkeitsbereiche von Objekten 108
GVL 93, 125, 194
Halt auf BP 163
Haltepunkt
  aktiv 240
  Aktivieren 557
  Aktuelle Trefferanzahl 557
  Alle aktivieren 557
  Alle deaktivieren 557
  Alle löschen 557
  AS 792
  Bedingung 557
  CFC Editor 623
  Darstellung 243
  Deaktivieren 557
  Eigenschaften 557
  inaktiv 240

Löschen 557
Neu 241
Position 241
Quellcodeposition 557
ST-Editor 794
Symbole 243
Haltepunkt 137
Haltepunkt setzen 240
Haltepunkt umschalten 240
Haltepunkt-Bedingung 557
Haltepunkte
Befehle 240
Haltepunkte 557
Haltepunkte-Dialog 557
Haltepunktposition
FUP/KOP/AWL Editor 742
Haltepunkt-Position 243, 557
Haltepunkt-Positionen 243
Handhabung einer mechanischen Bremnse 999
Hardware scannen 108, 287, 901
harter Umbruch 445
hasattribute 1078
hastype 1078
hasvalue 1078
Hauptnetzwerk 117, 118, 120
Hearbeat 936
Hearbeat Producer Time 936
Heartbeat Consumer 936
Heartbeat Producer Zeit 936
Hersteller ID prüfen 955
Herstellername 924
hexadezimal 441
H-Funktion 668
Hide 1062
Hide_all_locals 1063
Hierarchie von Applikationen 114
Hilfe
Index 384
Informationen... 385
Inhalt 384
Sprache 146
Suchen 385
Synchronisieren 385
Hilfe 146
Hilfe Befehle 384
Hilfe Menü 171
Hilfetext 495
Hintergrund 552
Hintergrund-Bild 104
Hintergrundfarbe für Textelement 454
Hintergrundfarbe für Trace-Aufzeichnung 825
HMI 181, 194, 196
Höchste Stationsadresse 978
Höhe beim CFC 256
Höhe des Operandenfeldes 461
Höhe des Operandenkommentarfeldes 461
Höhe des Operandensymbolfeldes 461
Höhe von AS-Schritten 467
Höhe Zelle in Seiten-Layout 485
homepage 579
Homepage 443
homing 999
Horizontal zentrieren 554
Horizontalausrichtung 485
Host
ENI Server 525
Host 510, 514
Host 561
Host Dateien 906
Hotkey 843
htm-Datei für Web-Visualisierung 193
htm-Seite 896
I 1169
i/o abbild 987
I/O Abbild
Adresse ändern 911

Adresse fixieren 911
I/O Abbild 911
I/O Abbild eines CANopen Moduls 947
I/O Abbild EtherCAT Master 954
I/O Abbild EtherCAT Slave 965
I/O configuration
local 1049
I/O configuration 1049
I/O Mapping in CANbus 946
I/O-Abbild von Profibus-Modulen 987
I/O-Behandlung 909
ICmpIoDrvDPV1C1Master 988
ICmpIoDrvDPV1C1Slave 988
ID für Abbildungen 104
Identifizieren eines Geräts 287
Ident-Nummer 982
IEC 61131-3 1090
IEC 61131-3 Erweiterung
ANY_TYPE 1091
LTIME 1091
Pointer 1093, 1107
Referenzen 1092, 1106
WSTRING 1091
Zeiger 1093, 1107
IEC 61131-3 Erweiterung 1091
IEC 61131-3 Sprache 463
IEC Objekte für Geräteinstanz 911
IEC operators 1111
IEC Sprachen 64
IEC-Adressierungsmodus 1169
IEC-Schritt 772
iecsfc.library 783, 784
IEC-Symbole der Bibliothek veröffentlichen 403
IEC-Zyklen zählen 818
IF 799, 802
IF instruction 802
IF-Anweisung 799
IIoDrvAcyclicServices.library 1180
IIoDrvBusControl.Library 1180
IIoDrvDPV1C1.library 1180
IIoDrvDPV1C2.library 1180
IIoDrvProfibus.Library 1180
Im Stop-Status E/A aktualisieren 909
image 839, 843
Image Element 104
Immer anhalten 241
Immer binden 561, 1066
Implementation
Haltepunkt setzen 241
implementieren 79
implementiert 73
Implementierungssprache für Methoden 73
IMPLEMENTS 79
implizite Variablen
AS 561
implizite Variablen in AS
Optionen 470
Implizite Variablen in AS 784
import 412
Import von V2.3 Projekt 338
Import vor Übersetzen 561
Import/Export Textlisten 530
Importiere PLCopenXML 415
Importieren 412
Importieren von Textlisten 98
In das aktuelle Format konvertieren 338
In Großbuchstaben umwandeln 527
In Kleinbuchstaben umwandeln 527
Inaktive Randfarbe 445
Index
Objektverzeichnis 939
Index 384
Index eines Bausteins 1117, 1158
INDEXOF 1093, 1107, 1117, 1131, 1157, 1158
info.cfg-Datei 108
information 993

Information
  Gerät 988
Information 930, 935, 946, 947
Information EtherCAT Master 954, 965
Information im Meldungsfenster 580
Information zum Gerät 924
Informationen zum Entwicklungssystem 385
Informationen... 385
Informationsausgabe 1055
Inhalt 384
Inhalt Seiten-Layout 485
Inhaltsoperator 1093, 1107, 1132
Inhaltsverzeichnis 484
Inhaltsverzeichnis Projektdokumentation 485
Inhaltsverzeichnis synchronisieren 385
INI 1051
init Methode 84
Init Modus 955
Init_Namespace 1064
Init_On_Onlchange 1065
Initialisierung
  Array 1097
  beliebige Ausdrücke 1035
  benutzerdefiniert 1034
  Funktionsbaustein 1064
  GVL 93, 1061
  Pragma 1061, 1065, 1069
  Standardwert 1034
  String-Variablen 1064
Initialisierung 936
Initialisierung 1025
Initialisierung 1034
Initialisierung bei Reset 430
Initialisierung von Funktionsblöcken 1051
Initialisierung von GVL-Variablen 93
Initialisierungs-Methoden 1051
Initialization
  Structures 1100
Initialization 1025, 1051
Initialize_on_call 1064
Initialschritt 501, 771, 784
init-Schritt 501
Inline monitoring
  FUP/KOP/AWL Editor 742
Inline-Monitoring
  Aktivieren 445
  CFC Editor 623
  Darstellung 445
  ST-Editor 794
Inline-Monitoring 445
Inline-Monitoring 794
Input Speicher 1169
inputs 843
Installation
  Bibliotheken 387
Installation aktualisieren 146
Installation and Profile Manager 62
Installation Befehle 392
Installation eines Geräts 293
Installation reparieren 146
Installieren
  Visualisierungselemente 392
Installierte Elemente 392
Instance-path 1065
Instanz
  Haltepunkt setzen 241
  Programmiersystem 338
Instanz 74
Instanz des EtherCAT Masters 954, 966
Instanz des EtherCAT Slaves 965, 967
Instanz eines Funktionsblocks 75
Instanz für Gerät 911
Instanz in Visualisierung 186
Instanzname 884, 1074
Instanzpfad
  Aufrufliste 559

für Haltepunktposition 241
Haltepunkt-Position 557
Instanzpfad über Pragma auf String 1065
INT 1088
INT_TO_BOOL 1135
integrierte Visualisierung 194
Intelligente Code-Bearbeitung 495, 497
intelligente Einrückung 445
Intelligentes Kodieren 452
Interface 84, 88, 1153
interior rotation 843
interne Bibliothek 139
Internes Kommand 843
Interpolationspunkte anzeigen 269
Intervall für Task 814
Invertieren in Trace-Aufzeichnung 825
invisibility of visualization elements 843
IoDrvCifXProfibus.library 987
IoDrvEtherCAT 966
IODrvEtherCAT.library 969
IP Adresse 287
IP-Adresse 901
IPM 62
isotropic 843
isotropic visualization elements 843
Jerk
CAM 588
Jerk 993
jitter 818
JMP 807
JMP Operator in AWL 713
JMP-Anweisung 807
K 657
Kaltstart 430
Kanal
Adresse 911
Kanal 911
Kanal für Geräte-I/Os 911
Kanal-Typ 911
Kanten in Layout 485
Kategorie einer Bibliothek 399
Kategorie für Bibliotheken 387
kein 259
Keine Initialisierung 936
Keine Verknüpfung 510
Keyboard 166, 482
Keypad 843
Kindapplikation 108
Kind-Applikation 114
Kindknoten 117
Klammer 527
Klammerung 445
Kleinbuchstaben 527
kleiner als 1128
kleiner als oder gleich 1128
Knoten zurücksetzen 936
Knotenadresse 120
Knoten-Name 901
Knotennummer 936
Knotennummer CANopen Gerät 936
Kommando
execute by visualization input 843
Kommandozeilenoption
culture 1024
Kommentar
FUP
KOP und AWL Editor 461
Variablendeklaration 497
Kommentar 897
Kommentar in Anweisungsliste 712
Kommentar in AS 771
Kommentar in FUP/KOP/AWL 733
Kommentar in ST 808
Kommentierung ein/aus 308
Kommentierung eines Netzwerks 308
Kommetar bei Auschecken 516

Kommunikation
Adressierung und Routing 118, 120
Netzwerk Topologie 117
Kommunikation mit Datenquelle 694
Kommunikation mit Steuerung 901
Kommunikationseinstellungen 901
Kommunikationshandbücher 52
Kommunikationsparameter 901
Kommunikationspfad zur Steuerung 306
Kommunikationspuffer Web-Visualisierung 896
Kommunikationstyp CANbus 936
Kompatibilitätsprofil 191
Kompilierung
Objekt 561
Kompilierung 134
Kompilierung 1078
Kompilierungsformat CAM 561
Kompilierungsformat Kurvenscheibe 561
Komponenten auflisten 452
Komponenten Auflisten Funktion 1087
Komponenten Auflisten Funktionalität 452, 474
Komponenten Auflisten für Texteingabe 98
Komponenten sofort nach Eingabe eines Zeichens auflisten 452
Komposition in CFC 616
Komprimieren 540
Konfiguration
Gateway 901
Scan 287
Trace 825
Konfiguration CAN-Bus 925
Konfiguration Datenquelle 697
Konfiguration des Elements Bar Display 204
Konfiguration des Elements Meter 201
Konfiguration des Elements Table 213
Konfiguration des Elements Trace 208
Konfiguration DP-Modul 985
Konfiguration eines Antriebs 993
Konfiguration eines DP-Moduls 987
Konfiguration eines Profibus DP 986
Konfiguration eines Visualisierungselements 843
Konfiguration Gerät 904
Konfiguration mit Hilfe des SoftMotion Geräte-Editors 993
Konfiguration Task 814
Konfigurationsabgleich 292
Konfigurationsabgleich für PROFINET IO Gerät 292
Konfigurationsdialog für Gerät 466
Konstante
Datum 1163
getypt 1166
Online 372, 1048
Uhrzeit 1164
Zeit 1162
Konstante 1048
Konstante 1162
konstante Eingänge in CFC 266
konstante Variable
Deklaration 497
konstante Variable 497
konstante Variable 1048
Konstanten
REAL und LREAL 1165
STRING 1166
Zahlen 1165
Konstanten ersetzen 372
Kontakt 319, 739
Kontakt einfügen 318
Kontakt einfügen rechts 319
Kontakt parallel einfügen oberhalb 320
Kontakt parallel einfügen unterhalb 319
Kontakte einfügen 320, 321
Kontaktplan 711
Kontaktplan-Ansicht 327
Kontinuitätsanforderungen Kurvenscheibe 561
Kontinuitätsprüfung Kurvenscheibe 561

Konverter Optionen 457
Konvertieren
Bibliotheksreferenz 338
Gerätereferenz 338
Konvertieren 354
Konvertieren zwischen FUP/KOP/AWL 326
Konvertierung 454
Konvertierung einer Bibliotheksreferenz 338
Konvertierung einer Gerätereferenz 338
Konvertierung in FUP/KOP/AWL 327
Konvertierung von Gerätereferenz 457
Konzepte in CoDeSys 57
Koordinaten Visualisierungseditor 843
Koordinatensystem für Trace-Aufzeichnung 825
Koordinatenverschiebung 664
KOP
Kontakt 739
Spule 740
KOP 711
KOP Befehle 730
KOP Editor Optonen 461
KOP-Ansicht 327
KOP-Editor
Cursor-Positionen 727
KOP-Editor 716
Kopfzeile 485
Kopieren 280
Kopieren in AS 768
Kopieren in AWL 720
Kopieren von Projekten 63, 354
Kreis 644
Kreis anhängen (gegen Uhrzeigersinn) 679
Kreis anhängen (im Uhrzeigersinn) 679
Kreis einfügen (gegen Uhrzeigersinn) 679
Kreis einfügen (im Uhrzeigersinn) 679
Kreisbogen 644
Kurve 839
Kurvenscheibe
Beispielapplikation 1008
Beispielapplikationen 1001
Eigenschaften 561
Grundeinstellungen 561
Kompilierungsformat 561
Kontinuität 561
wechselnde 1008
Kurvenscheibe 588
Kurvenscheiben-Antriebssteuerung
Beispielapplikation 1006
Kurvenscheiben-Antriebssteuerung 1006
Kurvenscheiben-Tabelle 588
Kurzform-Modus für Deklaration 1035
L 657, 783
L_AddLog 24
L_DM_ParameterAccess 25
L_IO1000_EPMS640_RS232Read 46
L_IO1000_EPMS640_RS232Write 49
L_SMC_AbortTrigger 33
L_SMC_AbortTrigger (für EtherCAT) 33
L_SMC_AbortTrigger_FastIO 42
L_SMC_FastOutput 43
L_SMC_FreeEncoder 45
L_SMC_ParameterAccess_FB 25
L_SMC_ReadAxisError 36
L_SMC_ReadParameter 34
L_SMC_ReadParameter (für EtherCAT) 34
L_SMC_SetTorqueLimit 27
L_SMC_SetTorqueLimit (für EtherCAT) 27
L_SMC_TouchProbe 29
L_SMC_TouchProbe_FastIO 38
L_SMC_WriteParameter 35
L_SMC_WriteParameter (für EtherCAT) 35
Label für Datenbankprojekt 524
Laden
remanente Variablen 1045
Laden 424
Laden auf Steuerung

mehrfach 427
Laden auf Steuerung 424
Laden einer Bibliothek 404
Laden und speichern 443
Ladeverhalten 443
Landessprache 528, 529, 1024
Länge der Anwenderparameter 982, 985
Länge der dargestellten Aufzeichnung 825
last jitter 818
latching 999
late link 139
Laufzeitsystemdateien generieren 250
Laufzeitsystem-Objekte 921
Layout
Projekteinstellungen 372
Layout für Projektdokumentation 485
Layout-Optionen für AS-Editor 467
LD Operator in AWL 713
LE 1128
LE Operator in AWL 713
Leeren Baustein einfügen 313
Leitungsverzweigung einfügen 324, 325
Leitungsverzweigung in FUP/KOP/AWL 737
Lesen von EtherCAT Parametern 970, 971, 973
Lesezeichen Befehle 238
Lesezeichen ein-/ausschalten 239
Lesezeichen löschen 239
Letze Version abrufen (Gesamtes Projekt) 516
Letzte Version abrufen (Selektierte Objekte) 516
Libraries menu 139
Life Time Factor 936
LIMIT 1126
Limitierung 1126
line 843
linearer Antrieb 993
linedirection 843
linestyle 843
linewidth 843
Linie 843
Linie anhängen 679
Linie einfügen 679
Linienstärke 843
Linientyp 843
linkalways 1066
Links ausrichten 553
LINT 1088
LN 1143
LoadAndWriteRecipe 758
LoadFromAndWriteRecipe 758
LoadRecipe 758
lock parameter 982
Log
Einträge löschen 908
Export und Import 908
Log 908
LOG 1144
Logarithmus 1143, 1144
Log-Datei exportieren 908
Log-Datei importieren 908
Logger im Laufzeitsystem 908
Logging offline 908
Login 424
Login-Dialog 134
Login-Dialog in Visualisierung 227
Login-Konfiguration Data Server 697
lokale Variabeln 1042
lokalisierter Text für Menütitel 479
Lokalisierung 1024
look variables 843
Löschen
Haltepunkt 557
Löschen 280
Löschen in AS 768
LREAL 1089
LREAL Konstanten 1165
LREAL_TO Konvertierungen 1137

LRW anstelle von LWR/LRD verwenden 951
LT 1128
LT Operator in AWL 713
LTIME 1091
LWORD 1088
M 1169
m_references 843
M4 Interface-Datei 250
M4-Schnittstellendatei für externe Bibliothek 250
MAC Adresse 287, 951
Makro anzeigen 507
Makro danach einfügen 507
Makro einfügen 507
Makro in AS 507, 508, 772
Makro verlassen 508
Makro-Editor 507, 508
mapping 1037
Mapping
  DP-Module 987
Mapping 911
Mapping hinzufügen 939
Mapping I/Os EtherCAT Master 954
Mapping I/Os EtherCAT Slave 965
Mapping von Bibliotheksreferenzen 338
Mapping von Gerätereferenzen 338
Mapping von PDOs 935
Marke für AS-Verzweigung 503
Marke in Anweisungsliste 712
Marke in AS
  Verzweigung 772
Marke in CFC 616
Marke in FUP/KOP/AWL 735
Master
  EtherCAT 951
Mausaktionen auf Visualisierungselement 843
Mausaktionen im Texteditor 445
MAX 1125
Max.Anzahl von Tasks 813
Max.Jitter 818
Max.Zykluszeit 818
Maximal aktiv 771
Maximalzeit für AS-Schritt 771
Maximumsfunktion 1125
MC_CAM_REF 588, 607, 608
Mehrbenutzerbetrieb 234
Mehrfacher Download... 427
Mehrfachkanal 540
Mehrsprachigkeit 98
Meldungen
  Gehe zu Quelltextposition 405
  Nächste Meldung 405
  Vorherige Meldung 405
Meldungen 163
Meldungen 580
Meldungen pro Task aktivieren 951
Meldungsausgabe erzwingen 1055
Meldungsfenster
  Pragma 1055
  Querverweise 555
  Übersetzung 248
Meldungsfenster 163, 247
Meldungsfenster 405
Meldungsfenster 580
Meldungskategorie 580
Meldungs-Pragma 1055
Meldungstyp 1055
Menü
  Standard 171
  Übersicht 171
Menü 171
Menü 479
Menü anpssen 166
Menübefehle 171
Menüleiste 171
Menüstruktur
  Einstellungen laden 479

Einstellungen speichern 479
Menüstruktur 479
Menütitel lokalisieren 479
Merkerspeicher 1169
meter element 201
Method 84
Methode
extern 92
zusätzliche Ausgänge 1042
Methode 77, 80, 84, 86
Methode 1051
Methode hinzufügen 84
Methode immer zyklisch aufgerufen 84
Methodenaufruf 80, 84
Methodenaufruf im Applikations-Stop 84
Methodenaufruf in AWL 720
Methodenimplementierungssprache 73
Methoden-Prototyp 88
M-Funktion 670
MIN 1126
Min.Jitter 818
Min.Zykluszeit 818
Minimal aktiv 771
Minimalzeit für AS-Schritt 771
Minimumsfunktion 1126
Mit Argumenten einfügen 68, 75, 495
Mit Download einloggen 134
Mit Namen 467
Mit Online Change einloggen 134
Mitgliedschaften 378
MOD 1115
Modi 666
Modifikator in AWL 713
Modulo Division 1115
Modulo-Einstellungen 993
monitoring
tasks 818
Monitoring
AS 792
CFC Editor 623
FUP/KOP/AWL Editor 742
ST-Editor 794
Texteditor 445
Tooltip 708
Überwachungsliste 898
Monitoring 137
Monitoring 708
Monitoring 1067
Monitoring Data Server 697
Monitoring-Format 441
Motion
Beispielapplikationen 1001
Motion 989
Motion 1001
MOVE 1115
MUL 1112
MUL Operator in AWL 713
multi-application 114
multi-device 108
multi-language support 98
Multiplexer Operator 1127
Multiplikation 1112
multi-resource 108
Multitasking 116
MUX 1127
N 783
N Modifikator 713
Nach Änderung in der Datenbank letzte Version abrufen 473
Nach Benutzer filtern 522
Nach Datenfluss anordnen 261
Nach erfolgreichem Übersetzen einchecken 473
Nach hinten legen 553
Nach oben suchen 329
Nach oben verschieben 281
Nach Öffnen eines Projekts letzte Version abrufen 473

Nach vorn legen 552
Nächste Anweisung anzeigen 246
Nächste Anweisung festlegen 245
Nächste Meldung 405
Nächster Editor 585
Nächstes Lesezeichen 239
Nächstes Teilfenster 586
Nachverschaltung 737
Name
  Bibliothek 748
  Gerät 924
  Projektdatei 335
Name einer Instanz 187
Namensfilter 329
Namenskonvention 1028
Namensraum
  Enumeration 1161
  Pragma 1064
  Projekt 366
Namensraum 470
Namensraum 1025
Namensraum 1159
Namensraum 1160
Namensraum ändern 403
Namensraum für Bibliothek 139, 403, 748, 1160
Namensraum-Operator 1159, 1160
Namensvergabe 1028
Navigation in Ansichten 413
Navigieren im Texteditor 238
NE 1130
NE Operator in AWL 713
Negation in FUP/KOP/AWL 321
negativer Durchlauf 588
Negieren in CFC 258
negierte Operation 713
Negierten Kontakt einfügen 319
Negierten Kontakt parallel einfügen unterhalb 319
negierter Kontakt 319
NetX
  Profibus 987
NetX Com Channel 987
NetX Konfiguration 987
Netz 924, 930
Netzwerk 307, 733
Netzwerk durchsuchen 306
Netzwerk einfügen 307
Netzwerk einfügen unterhalb 308
Netzwerk Topologie 117
Netzwerkadresse 120
Netzwerkeigenschaften 125
Netzwerkeinstellungen 125, 561
Netzwerkkommentar 733
Netzwerkkommentar anzeigen 461
Netzwerkkommunikation 125
Netzwerkmaske 298
Netzwerkpfad nicht im Projekt speichern 901
Netzwerkpfad setzen 901
Netzwerk-Scan 306, 901
Netzwerktitel anzeigen 461
Netzwerktyp 561
Netzwerkvariablen
  Eigenschaften 561
  Import und Export 561
  Verknüpfung mit Datei 561
Netzwerkvariablen 125
Netzwerkvariablenliste 93
Neue Aufzeichnung 825
Neue horizontale Registerkartengruppe 587
Neue Objekte mit Quellcodeverwaltung verknüpfen 473
Neue Symbolleiste 477
Neue Variable für Aufzeichnung 825
Neue vertikale Registerkartengruppe 587
Neuer Haltepunkt 241, 557
Neuer Menüpunkt 479
Neuer Separator 477, 479

Neues Gateway 901
Neues Gerät 901
Neues Popup-Menü 479
Neues Projekt 335
Neues Symbol 477, 479
Neuestes Profil verwenden 372
News Seite 443
nicht periodisch 598
Nicht verbundene Anschlüsse entfernen 267
Nicht verwendete FB-Aufruf-Parameter entfernen 326
Nicht verwendete Textlisteneinträge entfernen 535
no binding 561
No_check 1068
No_Copy 1068
No_virtual_actions 1070
Nocke 588
Nocke einfügen 588
Nockenschalter 588
Nockenspur 588
Node Guard 936
Node ID 936
Node-Guarding aktivieren 936
nodeguarding in CANbus 936
Node-ID 936
No-exit 1069
noinit 1069
normerweiternde Datentypen 1090
NOT 1119
Numpad 843
Nur aktive Transitionen berechnen 470
Nur anhalten falls... 241
Nur Bezeichnungen 520, 522
Nur ganzes Wort 329
nur tabellarisch 460
nur textuell 460
Nur Textunterschiede exportieren 530
NVL 93
Oben ausrichten 553
Obergrenze aktivieren für Trace-Variable 825
Objekt
  Dokumentation 485
  Übersetzungsoptionen 561
Objekt 64
Objekt 1006h unterdrücken 934
Objekt bearbeiten 408
Objekt bearbeiten - Offline 409
Objekt bearbeiten mit.... 408
Objekt exportieren 411
Objekt hinzufügen 407
Objekt importieren 412
Objekt öffnen mit 561
Objekt suchen 560
Objekt umbenennen 407
Objekte
  PLCopenXML 414
Objekte anzeigen 522
Objekte auf Gerät 921
Objekte aus der Quellcodeverwaltung hinzufügen 515
Objekte kopieren 354
Objekte überschreiben 354
Objekteigenschaften
  CAM 588
Objekteigenschaften 64
Objekt-Eigenschaften
  Allgemein 561
  AS-Einstellungen 561
  Bitmap 561
  Externe Datei 561
  Übersetzen 561
  Verknüpfung mit Datei 561
  Visualisierung 561
  Zugriffskontrolle 561
Objekt-Eigenschaften 561
Objektname 561
objektorientierte Programmierung 57, 59, 73, 77, 79, 80, 82, 84, 88

Objektorientierung 57
Objekttyp 366, 561
Objekttypen 329
Objekttypen im Projekt 57
Objekttypen Verfügbarkeit 463
Objektverzeichniseintrag wählen 939, 943
Obsolete 1072
Offline-Logging 908
Offline-Modus 169
Offline-Modus Ansicht 169
Öffnen
  Projekt 338
Öffnen mit
  Objekt 561
Öffnen von Fenstern 166
offset 598
Ohne Änderung einloggen 134
ondialogclosed 843
online
  Alle Forces zum Überwachen-Fenster hinzufügen 435
  Ausloggen aus Applikation 422
  datei von steuerung laden 426
  Einloggen auf Applikation 419
  Mehrfacher Download... 427
  Projekt von steuerung laden 426
  Quellcode von steuerung laden 426
  Sicherer Betrieb 417
  simulation von gerät 436
online 426
Online Befehle 417
Online Change
  remanente Variablen 1045
Online Change 134, 419, 424
Online Hilfe
  Sprache 146, 452
Online Hilfe Befehle 384
Online Menü 171, 417
Online Status auswählen 408, 409
Online Status der Applikation 163
Online-Ansicht 708, 794
Online-Benutzerverwaltung 917
Online-Modus
  ST-Editor 794
Online-Modus 169
Online-Modus Ansicht 169
Online-Status auswählen 169
onmousedown 843
onmouseenter 843
onmouseleave 843
onmousemove 843
onmouseup 843
Operand
  Optionen 461
Operand in AWL 720
Operand in Klammern in AWL 713
Operanden 1088
Operandenkommentar 716
Operandenkommentar anzeigen 461
Operational Modus 955
Operator
  Zuweisung 800
Operator 1111
Operator für Pragmas 1078
Operator in AWL 713, 720
opt.keyb-file 482
opt.tbar-file 477
Optimale Größe Web-Visualisierung 896
optional device default 934
Optionales Gerät 936
Optionen
  AS-Editor 467
  Bibliotheken 454
  CFC Editor 459
  CoDeSys 2.3 Konverter 457
  Deklarationseditor 460

Features 463
FUP
KOP und AWL 461
Geräteeditor 466
Intelligentes Kodieren 452
Laden und speichern 443
Quellcodeverwaltung 473
Syntaxhervorhebung 454
Visualisierungen 474
Optionen 442
Optionen 474
Optionen AS-Flags 470
Optionen für AS-Objekte 470
Optionen für Benutzeroberfläche 166
Optionen-Dialog 442
Options
Langage FBD
Langage LD et IL 461
Options SFC 470
OR 1118
OR Operator in AWL 713
Ordner 413
Ordner hinzufügen 413
Originaldatei öffnen 443
ORN 1118, 1159
Ort 514
output in FBD/set and reset 741
Output Speicher 1169
Owner 378
P 783
pack_mode 1073
Papier 485
Papierformat 485
Papierquelle 485
pApplicationInfo 84
Parabel 651
Parallel 502
parallel SDO action limit 934
Parallele Verzweigung 502, 503, 772
Parameter
Gerät 904
Typ 904
Parameter aktualisieren 310, 325, 735
Parameter bearbeiten in CFC 265, 266
Parameter dialog in Taskkonfiguration 818
Parameter für EtherCAT Master 953
Parameter für EtherCAT Slave 965
Parameter lesen 970
Parameter schreiben 973, 975
Parameter-Editor für CNC 693
parameterstringof 884
Parameterstringof 1074
PARAMS OF 71, 1051
Passwort
Benutzerkonto 378
ENI Server 525
Passwort 145, 179
Passwort 543
Passwort 917
Passwort für Projekt 372
Passwortbestätigung 917
Pathview-Editor 675
Pause in AS 784
PDO
aktivieren 938
Eigenschaften 939
PDO Eigenschaften 939
PDO Mapping 938, 943
PDO Mapping automatisch konfigurieren 935, 936, 939
PDO-Übertragung 936, 939
Periode Kurvenscheibe 561
Periodenübergang 561
periodisch 588, 598
PERSISTENT 497, 1045
persistente Variable

Deklaration 497
persistente Variablen 1045
Persistente Variablen 95
Pfad für Bilddateien 474
Pfad für Dateien für Textlisten 474
Pfeil für Trace-Aufzeichnung 825
Phys StartAdr 958
pins in CFC 267
Platzhalter
Visualisierung 186
Platzhalter 186
Platzhalter 187
Platzhalter 884
Platzhalter für Bibliothek 139, 399
Platzhalter für Seitengestaltung 485
Platzhalter für Seiten-Layout 485
Platzhalter in Menüs 479
Platzhalter in Symbolleisten 477
PLC Log 908
PLC Logger 908
PLC Service 147
PLCopenXML
Exportieren 414
import 415
PLCopenXML 414
plug-in 62
Pointer 1093, 1107
POINTER TO 1093, 1107
Poll timeout 978
Poly5 588
polygon 839, 843
polyline 843
Polylinie 843
Polynom 5.Grades 588
Port 510, 514, 561, 901
PosControl.project 1008
position 843
Position
Aufrufliste 559
Position für Haltepunkt 241
Position von Variablen 555
Position von Visualisierungselementen 842
Positionierung 640
Positionierung von Fenstern 166
positiver Durchlauf 588
Postfix in AWL 720
Potenzierung 1147
POU
Aufruf 66
Aufrufliste 559
für Haltepunktposition 241
Haltepunkt-Position 557
Suchen 246
POU 64
POU 66
POU 68
POU Eigenschaften 64
POUs
Übersetzen 249
POUs Ansicht 560
POUs for implicit checks 107
POUs für Task 814
POUs window 560
POUs-Fenster 64, 163, 560
Ppp Remote Address 298
Präfix in AWL 720
Pragma
Array für Visualisierung 1060
Attribut Global_init_slot 1061
Attribut Instance-path 1065
Attribut Monitoring 1067
Attribut No_check 1068
Attribut Reflection 1076
attributes 1065
Attribut-Pragmas 1056
bedingt 1078

Benutzerdefinierte Attribute 1057
define 1078
defined 1078
displaymode 1059
ExpandFully 1060
hasattribute 1078
hastype 1078
hasvalue 1078
Hide 1062
Hide_all_locals 1063
Init_Namespace 1064
Initialize_on_call 1064
linkalways 1066
No_Copy 1068
no_virtual_actions 1070
No-exit 1069
Noinit 1069
Pack_mode 1073
Parameterstringof 1074
Qualified_Only 1075
Symbol 1076
undefine 1078
xref 1078
Pragma 232, 561
Pragma 1056
Pragma für Datentyp 1072
Pragma für Meldungsausgaben 1055
Pragma für Symbolkonfiguration 1076
Pragma-Operatoren 1078
precompile 134
Precompile Information 372
Preoperational Modus 955
Printout 484
Priorität
Task 814
Tasks 818
Priorität 814
Produkt ID
EtherCAT Konfiguration 955
Produkt ID prüfen 955
Professional Feature-Set 57, 463
Profibus
Adresse 978
DP-Master 978
DP-Modul 985
DP-Parameter 985
DP-Slave 982
I/O Abbild 987
Profibus 987
PROFIBUS 52
Profibus DP Konfiguration 986
profibus i/o abbild 987
profibus i/o abbild master 987
Profibus Master
NetX 987
Profibus Master 987
Profibus Master 988
Profibus Slave 988
Profibus-Gerät
Information 988
Status 988
Profibus-Modulkonfiguration 987
Profil 62
Profil für Visualisierung 372
Profil für Visualisierungsprojekt 191
Profilauswahl 392
PROFINET IO
Konfiguration synchronisieren 292
Konfigurationsabgleich 292
PROGRAM 68
Program in ASCII-Datei schreiben 274
Program Organization Unit 66
Programm 68
Programm ausführen 909
Programm bei Gerätestop 909
Programm geändert 163

Programm geladen 163
Programm in FUP/KOP/AWL 735
Programm skalieren 272
Programm unverändert 163
Programm verschieben 271
Programm von ASCII-Datei laden 274
Programmaufruf 68, 814
Programmaufruf in AWL 720
Programmierobjekte 57
Programmiersprache
  CFC 614
Programmiersprache 711
Programmiersprache für Methoden 73
Programmiersprachen 64
Programmiersystem beenden 358
Programm-Name 1028
Programm-Organisationseinheit 66
Progrmm drehen 273
Projekt
  aus Quellcodeverwaltung 514
  Beispiel 147
  CoDeSys V2.3 63
  Konvertierung 338
  Kopieren 354
  Neu 335
  Öffnen 338
  Passwort 372
  Projektinformationen 366
  Schließen 345
  Sicherheit 372
  Sicherungskopie 338
  Speichern 346
  Speichern unter 347, 349
  Startseite 579
  Versionsgeschichte 522
  Vorlagen 335
  Zuletzt verwendete Projekte 358
  Zusammenfassung 366
Projekt 63
Projekt 359
Projekt als übersetzte Bibltiothek speichern 349
Projekt auf Steuerung laden 425
Projekt auf verbundene Steuerung schreiben 426
Projekt aus Quellcodeverwaltung öffnen 514
Projekt aus Steuerung laden 426
Projekt automatisch öffnen 443
Projekt dokumentieren 484
Projekt konvertieren 338
Projekt kopieren 63, 354
Projekt Menü 171
Projekt öffnen 338, 358, 514
Projekt schließen 345
Projekt speichern 346
Projekt speichern und ins Bibliotheks-Repository installieren 349
Projekt/Vergleichen 359
Projektarchiv
  Archiv speichern 349
Projektarchiv 349
Projektarchiv 352
Projektdatei 335
Projektdateien 347
Projektdatenbank
  Anstehende Eincheckvorgänge 582
Projektdatenbank 234, 473
Projekt-Dokumentation 237
Projekte verschmelzen 63
Projekteinstellungen
  Benutzer und Gruppe 372
  Benutzer und Gruppen 378
  Quellcode-Download 372
  Seiteneinstellungen 372
  Sicherheit 372
  Übersetzungsoptionen 372
  Visualisierung 372
  Visualisierungsprofil 372

Projekteinstellungen 63
Projekteinstellungen 372
Projekteinstellungen 378
Projektformat 338
Projektgeschichte
  Filter 520
Projektinformation 139
Projektinformationen
  Namensraum 748
Projektinformationen 63, 366
Projektkomprimierung 443
Projektname 335
Projektorganisation 57
Projektschutz 145
Projektstatistiken 366
Projektumgebung 338, 365
Projektvergleich 359
Projektverschlüsselung 372
Projektversion 338
Projektversionsgeschichte 522
Projektvorlage 335
Projekt-Vorlage 63
Prozedurschritt 244
Prozessdaten 959
Prozessdaten EtherCAT Slave 958
Prüfe Hersteller ID 955
Prüfe Produkt ID 955
Prüfsumme übertragen 561
pTaskInfor 84
Puffergröße für Trace-Aufzeichnung 825
Punkt einfügen
  CAM 588
Punkt einfügen 588
Q 1169
Quadratwurzel 1143
Qualified_Only 1075
Qualifizierer für AS-Aktionen 783
Qualifizierer in AS 505
Qualifiziererbreite 467
Qualität Ausdruck 485
Quelladresse MAC 951
Quellbibliothek 454
Quellcode auf Steuerung laden 425
Quellcode auf verbundene Steuerung schreiben 426
Quellcode aus Steuerung laden 426
Quellcode-Download 372, 425
Quellcode-Position finden 246
Quellcodeverwaltung
  Anmelden 525
  Anstehende Eincheckvorgänge 582
  Auschecken rückgängig 519, 582
  Ausschecken 516
  Bezeichnung 524
  Editor öffnen 582
  Einchecken 518, 582
  Letzte Version abrufen 516
  Objekte aus der Quellcodeverwaltung hinzufügen 515
  Optionen 473
  Projekt öffnen 514
  Projektversionsgeschichte 522
  Speicherort ändern 510
  Status aktualisieren 582
  Unterschiede anzeigen 582
  Unterschiede anzeigen... 519
  Unterschiede zu v2.3 236
  Verknüpfung 234
  Verknüpfung ändern 510
  Version zurücksetzen 520
  Versionsgeschichte 520
Quellcodeverwaltung 234
Quellcodeverwaltung 236
Quellcodeverwaltung 359
Quellcodeverwaltung 561
Quellcodeverwaltung 581
Quellcodeverwaltung ändern... 510

Quellcodeverwaltung für dieses Projekt verwenden 510
QuellcodeverwaltungeStatus aktualisieren 524
Quelle Papier 485
Quellgerät 457
Quelltextnavigation Befehle 246, 247
Querkommunikation 52
Querverweise ausgeben 247
Querverweise automatisch aktualisieren 452
Querverweise in Meldungsfenster 247
Querverweisliste
  Gehe zur Definition 247
Querverweisliste 247
Querverweisliste 452
Querverweisliste 554
Querverweisliste 555
R 259, 783
R operator 800
R Operator in AWL 713
Rahmen 843
Rahmen in Visualisierung 843
Rahmenattribute 843
Rahmenattribute für Visualisierungselemente 843
Rahmenfarbe 843
ramp_in 598
ramp_in_neg 598
ramp_in_pos 598
Randbereich Einstellungen 445
Ränder in Seiten-Layout 485
Raster anzeigen 270
Raster für Visualisierungseditor 474
ReadAndSaveRecipe 758
ReadAndSaveRecipeAs 758
ReadRecipe 758
REAL 1089
REAL Konstanten 1165
REAL to DINT Konvertierung 1141
REAL_TO Konvertierungen 1137
Reboot
  remanente Variablen 1045
Recht löschen 921
Rechte
  Zugriffsrechte auf Gerät 921
Rechte 179, 546
Rechte für Aktionen 546
Rechte für Objekt 561
Rechte für Zugriff auf Gerät 921
Rechteck 839, 843
Rechts ausrichten 553
Rechts einfügen danach 320
recipe 843
Recipe_Management.library 754
RecipeManCommands 754, 758
rectangle 843
references 843
Referenz
  __ISVALIDREF 1092, 1106, 1150
  Applikationen 905
  Bibliothek 403
  Datentyp 1092, 1106
  gültiger Wert 1092, 1106
  Prüfoperator 1092, 1106, 1150
  Visualisierung 186
Referenz 186
Referenz 884
Referenzen einer Variable 555
referenzierte Bibliotheken 139, 748
Referenzierte Bibliotheken konvertieren 338
Reflection 1076
Registerkartengruppe 587
regular expressions 329
Reguläre Ausdrücke verwenden 329
Reihenfolge 560
Reihenfolge von Visualisierungselementen 552, 553
Re-Initialisierung 430, 431
relativ 588, 598

relative Adresse 120
relative Adressierung 118
Relative Bewegung von Visualisierungselementen 843
relative movement 843
ReloadRecipe 758
remanente Variable
Deklaration 497
remanente Variablen 1045
Remanente Variablen löschen 441
Reparieren der Installation 146
REPEAT 806
REPEAT-Anweisung 799
REPEAT-Schleife 806
Repository 139
Repository bearbeiten 392
Repository für Bibliotheken 387
Repository für Geräte 293
Repository für Visualisierungselemente 392
Repository für Visualisierungslemente 392
Repository hinzufügen 392
Repository löschen 392
reset 259
Reset 323, 800, 1045
reset coil 741
Reset einer Spule 740
Reset für CANopen Knoten 936
Reset in AS 784
Reset Kalt 430, 1045
Reset Ursprung 431, 1045
Reset Ursprung Gerät 441
Reset Warm 1045
Reset Warm Applikation 430
Reset-Spule 317, 740
Reset-Spule einfügen 317
Ressource 114
Ressourcen-Objekte 57
RET Netzwerk 733, 742
RET Operator in AWL 713
RETAIN 497, 1045
Retain-Variable
Deklaration 497
Retain-Variablen 1045
Retry limit 978
Return 316
RETURN 799, 802
Return einfügen 316
RETURN in CFC 616
RETURN in FUP/KOP/AWL 736
Rezept 494
Rezeptur 754, 843
Rezeptur aus Datei laden 754
Rezeptur Befehle 494
Rezeptur einfügen 494, 754
Rezeptur entfernen 495
Rezeptur erstellen 754, 843
Rezeptur in Datei speichern 754
Rezeptur laden 843
Rezeptur lesen 754, 843
Rezeptur löschen 754, 843
Rezeptur schreiben 754, 843
Rezeptur speichern 843
Rezepturdefinition
Rezeptur einfügen 494
Rezeptur entfernen 495
Rezepturdefinition 754
Rezepturen entffernen 754
Rezepturname 754
Rezepturverwalter 494
Rezepturverwaltung 843
Richtung umkehren 274
ROL 1122
Rollwinkel der Kamera ändern 677
ROR 1123
rotation 843
Rotation 1122, 1123

Rotation von Visualisierungselementen 843
rotatorischer Antrieb 993
rounded rectangle 843
Router 901
Router für Gateway 298
Routing 118, 298
Rückgängig 445, 542
Rücksetzen von Anzeigeeinstellungen im Trace 825
Rücksetzen von Aufzeichnungseinstellungen 825
Runtime Error Codes 1180
S 163, 260, 783
S operator 800
S Operator in AWL 713
Safe Operational Modus 955
Satzunterdrückung 270
SaveRecipe 758
scaling 843
Scan der Hardware 108
Scannen der Hardware 287, 901
Schaltfläche 839, 843
Schaltpunkt 668
Schema 329
schittweise Ausführung
  Aufrufliste 559
Schleppfehler 993
Schleppfehlerüberwachung 993
Schließen
  Projekt 345
Schließen von Fenstern 166
Schlüssel
  Projektinformationen 366
Schlüsselwort 1045
Schlüsselworte automatisch in Großschreibung 452
Schlüsselwörter 1038
Schnittstelle
  erweitern 88
  hinzufügen 88
  Methoden 88
  Visualisierung 186
Schnittstelle 73, 77, 84
Schnittstelle 88
Schnittstelle 186
Schnittstellen implementieren 79, 409
Schnittstellen zu Profibus DP Master und Slave 988
Schnittstellen-Editor 186, 884
Schnittstellen-Editor in Visualisierung 551
Schreibe DCF 936
Schreiben 794
Schreiben von EtherCAT Parametern 973, 975, 976
schreibgeschützte Projekte bearbeiten 443
Schreibschutz 346
Schreibschutz umgehen 443
Schrift für Trace-Aufzeichnung 825
Schriftart für Dokumentation 485
Schriftart im Texteditor 445
Schriftart in Seiten-Layout 485
Schriftstil für Textelement 454
Schritt 137
Schritt aktiv 771, 772
Schritt aktiviert 771, 772
Schritt deaktiviert 771, 772
Schritt einfügen 501
Schritt in AS 501, 502, 772
Schrittaktion 772
Schrittattribute 467, 771
Schrittattribute in AS 792
Schrittbreite 467
Schritthöhe 467
Schrittname 771, 772, 784
Schrittstatus 784
Schritt-Transition danach einfügen 502
Schritt-Transition einfügen 501
schrittweise Abarbeitung 137
Schrittweise Abarbeitung in FUP/KOP/AWL 742
schrittweise Ausführung 240, 244, 245, 246
Schrittzeit 467, 784

Schrittzeit anzeigen 467
Scrollen 538
Scroll-Maus Zoomen im Trace 539
SD 783
SDO
  Abfolge 943
  Bearbeiten 943
  Hinzufügen 943
SDO action limit 934
SDO Aktionenbegrenzung 934
SDO Zugriffe 955
SDOs erzeugen 936
Seitenausrichtung 485
Seiteneinstellungen 372, 485
Seiten-Layout 485
SEL 1124
Selektieren im FUP/KOP-Editor 716
Selektieren von Visualisierungselementen 841, 843
Selektierte Anschlüsse verbinden 267
Selektionsfarbe für Trace-Aufzeichnung 825
Selektionsfarbe im Texteditor 445
Selektor 616
Sende PDO Mapping 943
Separator in Menü 479
SercosIII SCI xml erzeugen 285
Service Data Object 943, 959
set 260
Set 323, 800
set coil 741
set/reset in CFC 259, 260
Set/Reset in FUP/KOP/AWL 323, 740
Set/Reset in IL 741
Set/Reset outputs in FBD 741
set-Eigenschaft 86
SetRecipeManager 758
SetRecipeValues 758
Set-Spule 317, 740
Set-Spule einfügen 317
Setzen 800
SFC 766
SFC step 501
SFCActionControl 783
SFCCurrentStep 784
SFCEnableLimit 784
SFCErrorAnalyzationTable 784
SFCErrorPOU 784
SFCErrorStep 784
sfciec.library 470, 561
SFCInit 784
SFCPause 784
SFCReset 784
SFCTip 784
SFCTipMode 784
SFCTrans 784
Shift 1120, 1121, 1122, 1123
SHL 1120
shortcut keys 482
Shortcuts 482, 541
show frame 843
SHR 1121
Sicherer Online-Betrieb 417, 901
Sicherheit
  Projekteinstellungen 372
Sicherheit 145
Sicherheit 372
Sicherungskopie 338, 346
Sicherungskopie erzeugen 443
Sicherungskopie öffnen 443
Sichtbarkeit 497
Sichtbarkeit eines Visualisierungselements 843
Sichtbarkeit von Bibliotheksreferenzen 403
sigmoidal 993
Signaturen
  Applikationen 905
Simulation 108, 436
Simulation von Gerät 436

Simulationsgerät 108
Simulationsmodus 436
SIN 1144
sine 1144
singleton event 814
SINT 1088
Sinus 1146
SIZEOF 1116, 1158
SJA Chip 925
Skaleneinteilung für Trace-Aufzeichnung 825
Skalenunterteilung für Trace-Aufzeichnung 825
Skalieren/Mapping 993
Skalierung 598
Skalierung von Visualisierungselementen 843
SL 783
Slave im Profibus 982
Slaves automatisch neu starten 951
Slot für NetX Kommunikation 987
Slot in Gerätebaum 285
SM3_Basic.library 588, 607, 989
SM3_Basic.Library 1180
SM3_CNC.Bibliothek 1180
SM3_CNC.library 989
smart coding 1087
SMC_CAMTable 609, 610
SMC_CAMXYVA 609
SMC_CNC_REF 688
SMC_OUTQUEUE 688
SoftMotion
  Antriebskonfiguration 992
  Beispielapplikationen 1001
  Drive Interface 992
  Drive Logical 997
SoftMotion 989
SoftMotion Antrieb
  Skalieren/Mapping 993
SoftMotion Antrieb 992
Softmotion concept and survey on components 989
Softmotion Device Editor
  Übersicht über speziellle Anwendungsfälle 999
SoftMotion Drive Interface 992
SoftMotion Geräteeditor 993
Softmotion Konfigurationseditor 989
SoftMotion Konzept und Übersicht über Komponenten 989
Software-Endschalter 993
Sortieren nach 560
Sortierreihenfolge 560
Source Control commands 509
Spalte in Seiten-Layout 485
Spaltenüberspannung 485
spätes Binden 139
Speicher allozieren 1148, 1150
Speicheradresse 1169
Speicherbereich 1169
Speicherformat 338
Speichern der Trace-Konfiguration 540
Speichern eines Projekts 346, 347
Speicherort
  Datei 335
Speicherort ändern 510
Speicherort für Bibliotheken 387
Speicherort für Visualisierungselemente 392
Speicherort von Geräten 293
Speicherorte bearbeiten 392
Speicherprobleme 372
Sperrzeit PDO 939
Spezielle Sprache 452
Spezielle Verknüpfung 510
Spline 648
Spline anhängen 679
Spline einfügen 679
Sprachauswahl für Benutzeroberfläche 452
Sprachauswahl für Online Hilfe 146, 452
Sprachdatei 98
Sprachdateien für Visualisierung 474

Sprache
Menütitel 479
Umschalten über Visualisierungselement 843
Sprache 98
Sprache 528
Sprache 1024
Sprache einfügen 529
Sprache entfernen 529
Sprache in Visualisierung 474
Sprachumschaltung 843
Sprachumschaltung in Visualisierung 187
Springe zu Zeile bei Fehler 943
Springen zu Baustein 246
Sprung 314, 807
Sprung danach einfügen 506
Sprung einfügen 314, 506
Sprung in AS 506, 772
Sprung in AWL 713, 720
Sprung in CFC 616
Sprung in FUP/KOP/AWL 735
Sprünge 657
Sprungmarke 314, 506, 807
Sprungmarke einfügen 316
Sprungmarke in AS 772
Sprungmarke in FUP/KOP/AWL 316
Sprungmarke in Netzwerk 316
SPS-Einstellungen 909
SPS-Logik 108
Spule 309, 317, 740
Spule einfügen 317
Spule setzen 740
Spule zurücksetzen 740
Spule/Set und Reset 740
Spur 588
SQRT 1143
ST
Anweisung 799
Ausdruck 798
erweitert 797
Kommentar 808
Strukturierter Text 797
Zuweisung 799
ST Operator in AWL 713
Standard Feature-Set 57, 463
Standard.Library 1180
Standardansichten 560
Standardbefehle 171
Standardmenüs 171
Standard-Namensraum 366
Standardrecht 179
Standardverknüopfung 561
Standardverknüpfung 510
Standardwert eines Geräteparameters 993
Standardwert für Geräteparameter 904
Start Applikation 427
Start/Stop Trace 537
Starten mit kleinem Beispielprojekt 147
Startseite 443, 579
Startüberprüfung 955
Startverhalten 443
Startvisualisierung 186, 561, 895, 896
Start-Visualisierung 193
state variables 843
static id 843
static ID 104
Stationsadresse 978, 982
Stations-Alias 955
Stationsname 287
statische Abbildung 104
Statische Texte 187
statische Variablen 1044
statischer Text 98
Statistiken
Projektinformationen 366
status 993
Status

Profibus-Gerät 988
Status 930, 935, 946, 947, 954, 965
Status aktualisieren 524, 582
Status des EtherCAT Slaves 967
Status einer Task 814, 818
Status Quellcodeverwaltung 582
Status von AS-Aktionen 784
Status von AS-Schritten 784
Status-Informationen Gerät 924
Status-Variablen für Visuailsierungselement 843
Statuszeile 163
ST-Code ausführen 843
ST-Code über Visualisierungselement ausführen 843
Steckplatz in Gerätebaum 285
ST-Editor
Haltepunkt-Positionen 794
JMP 807
Monitoring 794
ST-Editor 794
steigende Flanke 322
Stepping
Aufrufliste 559
Ausführen bis Cursor 245
Ausführen bis Rücksprung 245
Einzelschritt 244
Nächste Anweisung anzeigen 246
Nächste Anweisung festlegen 245
Prozedurschritt 244
Stepping 137, 240
Steuerung
Datei-Transfer 906
Logger 908
Steuerung als Dienst 147
Steuerung zurücksetzen 441
Steuerungskonfiguration 117
Steuerungstechnik 52
Stop Applikation 427
Stop Trace 537
String
Pragma 1065
STRING 1089, 1092, 1106
STRING_TO Konvertierungen 1140
STRING_TO_BOOL 1135
STRING-Konstanten 1166
String-Länge im Inline-Monitoring 445
String-Variablen
Initialisierung 1064
Stromleitung 711
STRUCT 1100
Struktur
Packen 1073
Struktur 1167
Strukturen
Initialisierung 1100
Strukturen 1100
Strukturierte Ansicht
Datenquelle 697
Strukturierte Darstellung 495
Strukturierter Text 797
Strukturvariable in CFC 616
SUB 1113
SUB Operator in AWL 713
Subindex
Objektverzeichnis 939
Subnetz Index 118
Subnetzindex 120
Subnetzmaske 287
Subnetzwerk 117, 118, 120
Subtraktion 1113
Suche Quellcode-Position 246
Suche Quelltext 246
Suchen 333, 384, 385
Suchen nach 329
Suchen nach Selektion 329
Suchen/Ersetzen 328
Suchraum 329

Suchschema 329
Suchstring 329
SUPER 77
suppress 1006h 934
Symbol 232, 1076
Symbol in AS 771
Symbol suchen 246
Symboladresse anzeigen 461
Symboldatei 232
Symbolische Werte 982
symbolische Werte DP-Modulparameter 985
Symbolkommentar 716
Symbolkommentar anzeigen 461
Symbolkonfiguration
  Pragma 1076
  xml-Datei 232
Symbolkonfiguration 232
Symbolkonfiguration 771
Symbolkonfiguration 1076
Symbolleiste
  Einstellungen laden 477
  Einstellungen speichern 477
Symbolleiste 163, 166
Symbolleiste 477
Symbolleiste bearbeiten 477
Symbolliste 1076
Sync 0 955
Sync COB-ID 936
Sync Manager 958
Sync Zykluszeit 955
synchrone PDOs 939
synchrone Übertragung 939
Synchronisation in EtherCAT 955
Synchronisationsmanager 958
Synchronisieren 385
Sync-Modus 978
Syntaxbäume in Datei speichern 372
Syntaxhervorhebung 454
SysCom.library 1180
SysComAsync.library 1180
SysCPUHandling 1180
SysDir.library 1180
SysDirAsync.library 1180
SysEvent.library 1180
SysFile.library 1180
SysFileAsync.library 1180
SysInt.library 1180
SysMem.library 1180
SysPci.library 1180
SysPort.library 1180
SysPortAsync.library 1180
SysProcess.library 1180
syssem.library 116
SysSem.library 1180
SysSemProcess.library 1181
SysShm.library 1181
SysSocket.library 1181
SysSocketAsync.library 1181
SysTask.library 1181
Systemaufrufe ermöglichen 561
Systembibliotheken 1180
Systembibliotheken verstecken 463
Systemereignisse 818
Systemzeit 187
SysTime.library 1181
SysTimer.library 1181
SysTypes.library 1181
t# 1162
tabellarischer Deklarationseditor 281, 460
Tabelle für Array-Visualisierung 213
Tabelle in Visualisierung 213, 843
Tabellendarstellung 673
Tabelleneditor 673
Table 213, 843
Tabs beibehalten 445
Tabulator beibehalten 445

Tabulatorgröße 445
TAN 1145
Tangens 1147
tangent 1145
tap false 843
tap on enter if captured 843
tap variable 843
Target-Visualisierung
  Automatische Anpassung 895
  Editor 895
Target-Visualisierung 181, 192
Target-Visualisierung 895
Targetvisualisierung öffnen 561
TargetVisualization 192
Task
  Abarbeitungsreihenfolge 818
  Anzahl Zyklen 818
  Aufrufliste 559
  Haltepunkt 557
  jitter 818
  Konfigurationsdialog 814
  Max.Anzahl 813
  monitor 818
  Parameter 818
  POU zuweisen 814
  Priorität 818
  Reihenfolge 814
  status 814, 818
  Trace 825
  Zykluszeit 818
Task für Trace-Aufzeichnung 825
Task-Editor
  Online-Ansicht 818
Task-Eigenschaften 814
Taskkonfiguration
  Eigenschaften 813
  Max.Anzahl von Tasks 813
  Monitoring 818
  Parameter 818
  Systemereignisse 818
Taskkonfiguration 116
Taskliste 814
Taskname 814
Tastaturbedienung
  Visualisierungseditor 841
Tastaturbedienung 482, 551
Tastaturbedienung aktivieren 554
Tastaturbedienung der Visualisierung 219
Tastaturbedienung in Visualisierung 554
Tastatur-Ereignisse bemerken 220
Tastaturfunktionen anpassen 166
Tastaturkonfiguration
  Editor 888
Tastaturkürzel
  Speichern 482
Tastaturkürzel 482
Tastaturkürzel für Visualisierungslement 843
Taste 888
Tasten 482
Tastenkombinationen 541
Taufen 287
tBit 978, 982
temporäre Variablen 1044
Text 187, 1055
Text für Visualisierungselement 841
Text in Seiten-Layout 485
Text in Visualisierung 187
text properties 843
Text und Sprache in Visualisierungen 187
Textbereich Einstellungen 445
Textdatei für textuelles 474
Textdatei in Editor einfügen 526
Texteditor
  Farben 445
  Lesezeichen 238
  Optionen 445

Randbereich 445
Schriftart 445
Texteditor 633
Texteditor Befehle 526
Texteigenschaften 843
Texteigenschaften für Visualisierung 843
Texteigenschaften Visualisierung 187
Textfarbe 454
Textfeld 843
Textfield in Visualisierung 843
Textformatierung 98, 187
textlist 187
Textliste 98, 187, 529
Textliste exportieren 530
Textliste importieren 530
Textlisten 474, 528
Textlisten-Dateien 372, 474
texts 843
textueller Deklarationseditor 460
Textvariable 187
Textvariablen 843
THIS pointer 84
THIS Zeiger 84
TIME 1090
Time Data Types 1090
TIME Funktion 1172
TIME_OF_DAY 1090
TIME_OF_DAY Konstanten 1090, 1164
TIME_OF_DAY Konvertierungen 1138
TIME_TO Konvertierungen 1138
TIME_TO_BOOL 1135
TIME-Konstanten 1162
Timeout in AS 771, 784
Tip-Modus in SFC 784
Titel 366
TO_BOOL Konvertierungen 1135
TOD 1090
tod# 1164
TOD_TO conversions 1138
Toggeln einer Variable 843
Toolbars 477
Toolbox
CAM 588
Toolbox 583
ToolBox
Visualisierungselemente 392
ToolBox 163
Toolbox für FUP/KOP/AWL 731
Toolbox in SFC 772
Toolbox Visualisierung 839
tooltip 843
Tooltip im FUP/KOP/AWL-Editor 716
Tooltip Text 98
Topologisch Anordnen 264
TQUI 978
Trace
Anzeige zurücksetzen 539
Anzeigeeinstellungen 825, 834
Aufzeichnungseinstellungen 825
Außerhalb des Anzeigebereichs 541
Cursor 537
Darstellung 825
Darstellung der neuen Variable 823
Dehnen 539
Farbe der neuer Variable 823
Komprimieren 540
Konfiguration der neuen Variable 823
Konfiguration laden 540
Mehrfachkanal 540
Obergrenze der neuen Variable 823
Scrollen 538, 541
Start/Stop 537
Tastenkombinationen 541
Trace einloggen 536
Trace laden... 540
Trace speichern... 540

Trigger zurücksetzen 537
Untergrenze der neuen Variable 823
Variableneinstellungen 825
Zoomen 539, 541, 820
Trace 208, 536
Trace Befehle 536, 540
trace commands 539
Trace Editor 536
Trace einloggen 536
Trace Kommando 540
Trace Konfiguration 825
Trace Shortcuts 541
Trace speichern... 540
trace-Datei 540, 820
Trace-Editor
Online-Modus 835
Trace-Editor 820
Trace-Fenster 820
Trace-Konfiguration aus Datei laden 540
Trace-Konfiguration speichern 540
Trace-Konfigurationsdatei 540
Trace-Menü 820
Trace-Puffer
Außerhalb des Anzeigebereichs 538
Trace-Puffer 820
Trace-Variable 820
Trace-Variable definieren 825
Transition
Code-Generierung 470
Mehrfachverwendung 772
Transition 501
Transition 502
Transition 772
transparent 843
transparent color 843
Transparenzfarbe 561
trapezoid 993
Trefferanzahl für Haltepunkt 241
Treiber 901
Treiber-Typ 901
Trennen von Gerät 441
Trennlinie in Menü 479
Trigger 820
Trigger Flanke für Trace-Aufzeichnung 825
Trigger für Trace-Aufzeichnung 825
Trigger Level für Trace-Aufzeichnung 825
Trigger Position für Trace-Aufzeichnung 825
Trigger Variable 825
Trigger zurücksetzen 537
TRUNC 1141
TRUNC_INT 1141
TSDR 978, 982
TSET 978
TSL 978
TTR 978
txtrecipe 754
Typ eines Parameters 904
TYPE 1091
typed literals 1048
Typed Literals 1166
ÜB 163
Überprüfe alle Pool Objekte 249
Überschreibmodus 526
Überschritenebene 485
Übersetungs-Eigenschaften 561
Übersetzen
Applikation 248, 249
Compiler-Definition 561
Externe Implementation 561
externe Implementierung 561
Immer binden 561
Objekt 561
Optionen für AS-Objekte 470
Pool Objekte überprüfen 249
POUs 249
Systemaufrufe ermöglichen 561

Vom Übersetzen ausschließen 561
Übersetzen Befehle 248
Übersetzungsfehler 249
Übersetzungsinformation 252
Übersetzungsmeldungen 248
Übersetzungs-Operatoren 1078
Übersetzungsoptionen 372
Übersetzungsoptionen für AS 470
Übersetzungsprozeß 134
Übersetzungsprozess 1078
Übersicht über speziellle Anwendungsfälle 999
Übertragung bei Änderung 561
Übertragung von Dateien 906
Übertragungstyp PDOs 939
Überwachen
Anlegen einer Überwachungsliste 897
Überwachungsliste im Online-Modus 898
Überwachen 584
Überwachen-Ansicht 584
Überwachung 163
Überwachung CANopen Gerät 936
Überwachung einer Task 116
Überwachungsfenster 163, 898
Überwachungsfenster CANbus 936
Überwachungsfunktionen
CheckPointer 1093, 1107
Überwachungsliste 584
Überwachungssteuerung 982
UDINT 1088
Uhrzeit 1164
UINT 1088
ULINT 1088
Um Eins nach hinten legen 553
Um eins nach vorn legen 552
Umbruchstelle 445
umschaltbar zwischen textuell und tabellarisch 460
Umschaltung 598
Unbekannte Variablen automatisch deklarieren 452
undefine 1078
Undocking views 166
ungarische Notation 1028
Ungleichheit 1130
Unicode 98
UNION 1091
Unlock parameter 982
Unsichtbarkeit eines Visualisierungselements 843
Unten ausrichten 553
Unterbereichstypen 1103
Untergrenze aktivieren für Trace-Variable 825
Unterschiede anzeigen 518, 582
Unterschiede anzeigen... 519
Unterschiede zu CoDeSys V2.3 59
Unterschiede zwischen Datenbankobjektversionen 520
Unterstützte Programmiersprachen 64
Unverankert 166, 585
Update 146
Update der Installation 146
Ursprung beim CFC relativ verschieben 256
ursprüngliches Gerät 457
USINT 1088
Util.library 1180
V2.3 Projekt importieren 338
V2.3 Projekte 63
VAR 1042
VAR CONSTANT 1048
VAR PERSISTENT 95, 1045
VAR RETAIN 1045
VAR_CONFIG 1049
VAR_EXTERNAL 1044
VAR_GLOBAL 1043
VAR_IN_OUT 267, 1043
VAR_INPUT 1042
VAR_INPUT CONSTANT 266
VAR_OUTPUT 1042
VAR_STAT 1044

VAR_TEMP 1044
Variable
  Ausgang 1042
  Bitzugriff 1167
  Ein- und Ausgang 1043
  Eingang 1042
  extern 1044
  global 1043
  lokal 1042
  Positionen im Projekt 555
  Querverweis 247
  statisch 1044
  temporär 1044
  Verwendungsstellen 247
  Zugriff 1167
Variable 1025
Variable 1167
Variable deklarieren 497
Variable einfügen in Rezeptur 754
Variable entfernen aus Rezeptur 754
Variable schreiben 843
Variable schreiben in Visualisierung 220
Variable toggeln 843
Variable umschalten 843
Variablen
  implizite in AS 470
Variablen in Eingabehilfe verstecken 1062, 1063
Variablen packen 561
Variablen setzen 278
Variablenaustausch 125
Variablendeklaration
  Attribute 1045
  Dialog 497
  unvollständige Adresse 1049
Variablendeklaration 1025
Variablendeklaration 1035
Variableneinstellungen 825
Variablengröße 1116, 1158
Variablen-Initialisierung 1034, 1035
Variablenkonfiguration 1049
Variablenlistenkennung 561
Variablenname 1028
Variablenverwendung 555
Variablenwert 708, 897
Variablenwerte schreiben 754
Variablenzugriff über Symbol 232
Vaterapplikation 108
Vater-Applikation 114
Vaterknoten 117
Verankern 166
verbieten 921
Verbinden mit Gerät 441
Verboten 546
Vererbung 77, 79, 80, 82
Vererbung von Rechten 179
Verfügbare Elemente 392
Verfügbare Geräte 287
Verfügbare Variablen 697
verfügbare Visualisierungen 550
Vergleichen 359
Vergrößern 677
Verhalten der Ausgänge bei Stop 909
Verkleinern 677
Verknüpfung 234
Verknüpfung ändern 510, 561
Verknüpfung mit Datei 561
Verknüpfung mit Projektdatenbank
  Optionen 510
verschachtelte Kommentare 808
Verschiebung 1120, 1121, 1122, 1123
Verschlüsseln einer Bibliothek 349
Verschlüsselungspasswort 372
Version
  andere Projektversion öffnen 338
  Bibliothek 399
  immer neueste 403

Projektdatei 338
Projektinformationen 366
Version 62
Version einer Bibliothek 139, 748
Version Programmiersystem 347
Version zurücksetzen 520
Versionsaktualisierung 338
Versionsdetails für Dateinbankobjekt 520
Versionseinschränkung 62
Versionsgeschichte 520
Versionsgeschichte gesamtes Projekt 522
Versionsgeschichtedetails 520, 522
Versionshandling für Bibliothek 403
Versionskompatibilität 365
Versionsverwaltung 57, 234
versteckte Bibliotheken 139
Verteilte Uhren beim EtherCAT Slave 955
Vertikal zentrieren 554
Vertikalausrichtung 485
Verwendung von Variablen 672
Verwendungsstellen von Variablen 247, 555
Verzeichnis für Bilddateien 474
Verzeichnis für Textlisten 474
Verzugszeit 647
Verzweigung
alternative 772
parallel 772
Verzweigung 503
Verzweigung einfügen 503
Verzweigung in AS 502, 503, 772
Verzweigung in FUP/KOP/AWL 324, 325, 737
Verzweigung rechts einfügen 503
Verzweigungsmarke 503
Verzweigungs-Marke 772
Vielsprachigkeit 98
View 163
virtueller Modus 993
visu_usewstring 98
Visualiseirung
Tastaturbedienung 554
Visualiserungselemente-Repository 392
Visualisierung
Bild 552
Bilddateien 372
Bitmap 552
Darstellungsgröße 561
Datenquelle 697, 704
Edit Control 220
Eingabe-Dialoge 227
Element ID 888
Elementliste 552
externes Gerät 196
Farbe 552
Frame Auswahl 550
Größe anpassen 474
Gruppe 549
Gruppierung aufheben 549
Hintergrund 552
HMI 196
integriert 194
Konfiguration des Elements Bar Display 204
Konfiguration des Elements Meter 201
Konfiguration des Elements Table 213
Konfiguration des Elements Trace 208
läuft auf Zielsystem 192
läuft im Programmiersystem 194
Objekt-Eigenschaften 561
Optionen 184
Projekteinstellungen 372
Raster 474
Repository 392
Schnittstellen-Editor 551, 884
Tabelle 213
Tastaturbedienung 219, 551
Tastatur-Ereignisse bemerken 220
Tastaturkonfigurations-Editor 888

Text für Element 841
Textlisten 98
Textlisten-Dateien 372
Überblick 181
Umschalten zwischen 550
Visualisierung eines Trace 208
Visualisierungselemente vom Typ Visualisierung 198
Voraussetzungen 184
Visualisierung 181
Visualisierung 181
Visualisierung auf externen Geräten 196
Visualisierung ausdrucken 843
Visualisierung bearbeiten 185
Visualisierung eines Trace 208
Visualisierungen
Optionen 474
Visualisierungsbibliotheken 191
Visualisierungseditor
Optionen 474
Schnittstellen-Editor 551, 884
Tastaturbedienung im Editor 841
Visualisierungseditor 548
Visualisierungseditor 884
Visualisierungs-Editor
Elementliste 890
Tastaturkonfiguration 888
Visualisierungs-Editor 185
Visualisierungselement
Ausrichtung 553, 554, 842
Auswahl 219
Eigenschaften 843
Größe 842
Gruppierung 842
Hintergrund 842
Hotkey 843
Konfigurieren mit Datenquellenvariablen 704
Position 842
Reihenfolge 552, 553, 842
Typ 843
Vordergrund 842
Visualisierungselement einfügen 549, 839
Visualisierungselement konfigurieren 843
Visualisierungselemente
Aktualisierung 392
Bibliothek 392
Eigenschaften 583
Elementeigenschaften 583
Gruppieren 549
Gruppierung aufheben 549
Installation 392
Installierte 392
Verfügbare 392
Visualisierungselemente 843
Visualisierungselemente 890
Visualisierungselemente auswählen 554
Visualisierungselemente selektieren 841
Visualisierungselemente vom Typ Visualisierung 198
Visualisierungselemente-Repository 191, 392
Visualisierungshintergrund 104
Visualisierungsinstanz 186
Visualisierungskommandos 548
Visualisierungsname 1028
Visualisierungsobjekt
Anlegen 185
Visualisierungsobjekt 185
Visualisierungsprofil
Anlegen 392
Name 392
Visualisierungsprofil 185, 191, 338, 365, 372
Visualisierungsprofil 392
Visualisierungsreferenz 186, 884
Visualisierungs-Schnittstelle 186
Visualisierungs-Template
Beispielapplikation 1005
Visualisierungs-Template 1005

Visualisierungs-Text-IDs aktualisieren 535
Visualisierungs-Text-IDs prüfen 535
Visualisierungsverzeichnisse 372
Visualisierungswechsel 843
Visualization Editor 548
Visualization Manager 185
VisuDialogs.library 191, 227
VisuElemBase.library 191
VisuElemFunctionality.library 191
VisuElems.library 191
VisuExternals.library 191
VisuInputs.library 191
VisuInternCommands.library 191
VisuManager 227
Vollbildmodus 579
Volltextsuche in Online Hilfe 385
vom Gerät laden 917, 921
Vom Übersetzen ausschließen 561
von der Platte laden 917, 921
Von DXF-Format importieren 275
Vor dem Editieren auschecken 473
Vor Speichern eines Projekts einchecken 473
Vor Übersetzen letzte Version abrufen 473
vorbereiteter Wert 431, 708
Vordergrundfarbe für Textelement 454
Voreinstellungen
  Objekt-Eigenschaften 561
Voreinstellungen 561
Voreinstellungen DP-Master 978
Vorherige Meldung 405
Vorheriger Editor 585
Vorheriges Lesezeichen 239
Vorheriges suchen 333
Vorheriges suchen (Auswahl) 333
Vorheriges Teilfenster 586
Vorlage 63
Vorlagen
  Kategorien 335
  Projekt 335
Vorschau für Projektdokumentation 485
Vorverarbeitete Bahn anzeigen 269
Vorverarbeitung 661
W 1169
Wählen Sie einen Eintrag aus dem Objektverzeichnis 943
Warmstart 430
Warnung 580, 1055
Warnung für Datentyp 1072
watch expression 708
watchdog
  task 814
Watchdog DP-Slave 982
Watchdog für Tasks 814
watchlist
  alle forces hinzufügen 435
Web-Browser 193, 896
Web-Server 193
webvisu.htm 193
Web-Visualisierung
  Editor 896
  Konfiguration 193
  Optimale Größe 896
Web-Visualisierung 181
Web-Visualisierung 193
Web-Visualisierung 896
Wechsel zwischen Kontrollmodi 999
weicher Umbruch 445
Weite beim CFC 256
Weitersuchen 329, 333
Weitersuchen (Auswahl) 333
Weiterverschaltung festlegen 323
Weiterverschaltung in FUP/KOP/AWL 323
Werkseinstellungen 936
Werkzeugbox
  CAM 588
Werkzeugbox in AS 772

Werkzeuge 583
Werkzeuge für FUP/KOP/AWL 731
Werkzeuge für Visualisierung 839
Werkzeuge in Visualisierungseditor 839
Werkzeuge Menü 171
Werkzeugleiste 163, 166
Wert eines Geräteparameters 904
Wert vorbereiten 432
Wert zurücksetzen 432
Werte auf Steuerung schreiben und lesen 754
Werte auf Variablen schreiben 754
Werte forcen
  Überwachungsliste 898
Werte forcen 432, 794
Werte schreiben
  Überwachungsliste 898
  Wert vorbereiten 432
Werte schreiben 431
Werte schreiben 794
WHILE 805, 807
WHILE-Anweisung 799
Wiederherstellen 542
WORD 1088
Word-Adressierung 1169
Wortumbruch 445
WriteRecipe 758
writing values 794
WSTRING 1091
X 1169
X/Y-Position in Visualisierung 843
X-Achse in Trace 825
XML-Datei für Geräte-Logging 908
xml-Datei für Symbolkonfiguration 232
xml-Export von Objekten 411
X-Offset beim CFC 256
XOR 1119
XOR Operator in AWL 713
XORN 1119, 1159
xref 1078
Y-Achse in Trace 825
Y-Offset beim CFC 256
Z 163
Zahlenkonstanten 1165
Zei 163
Zeichensatz 843
Zeige nächste Position 555
Zeige Position 555
Zeige vorige Position 555
Zeiger 1093, 1107
Zeile
  Endmarkierung 445
  Farben 445
Zeile im Texteditor 526
Zeile in AWL einfügen 321
Zeile in AWL löschen 321
Zeile in Seiten-Layout 485
Zeilen nummerieren 445
Zeilenendemarkierung 445
Zeilennummer 445
Zeilenüberspannung 485
Zeit eines AS-Schrittes 467
Zeitattribute in AS 771
Zeitdatentypen 1090
Zeiten in AS 771
Zeitstatus eines AS-Schrittes 784
Zeitsynchronisation mit Interpolator 666
Zeitüberschreitung in AS 771
Zeitüberwachung 814
Zeitüberwachung EtherCAT 955
Zeitüberwachung in AS 771
Zieladresse MAC 951
Zielbibliothek 454
Zielsystem
  Datei-Transfer 906
Zielsystem 108
Zielsystem auswählen 457

Zielsystembibliothek setzen 454
Zielsystem-ID 901
zielsystemspezifische Bibliothek 139
zielsystemspezifische Bibliotheksreferenz 399
Zielsystem-Typ 901
zoom 166
Zoom
  Trace 539
Zoom zu Baustein 246
Zoomen 539
Zugangsdaten ENI Server 525
Zugangsdaten Quellcodeverwaltung 525
Zugangsdaten speichern 525
Zugriff auf Variable
  Querverweisliste 555
Zugriff auf Variablen 1076
Zugriff auf Variablen über Symbol 232
Zugriff erlauben 921
Zugriff verbieten 921
Zugriffskontrolle
  Objekt 561
Zugriffskontrolle 145, 179
Zugriffskontrolle 561
Zugriffsrecht
  Datenquellenvariablen 697
  Symbol 1076
Zugriffsrecht 543, 546
Zugriffsrechte 179, 561
Zugriffsrechte auf Gerät 921
Zugriffsrechte online 921
zuletzt verwendete Dateien 443
Zuletzt verwendete Projekte 358
zum lokalen Gateway verbinden 298
Zurücksetzen 800
Zurücksetzen Task-Analyse 818
Zurücksetzen Taskwerte 818
Zurücksetzen von Zugriffsrecht 921
Zusammenfassung
  Projektinformationen 366
Zusammengehörige Klammern 445
zusätzliche Dateien 349, 352
Zuweisung 799, 1115
Zuweisung als Ausdruck 798
Zuweisung einfügen 309
Zuweisung in FUP und KOP 309
Zuweisung in FUP/KOP/AWL 734
Zuweisung in KOP 740
Zuweisungsoperator 800
Zwischenablage Befehle 279
zyklisch 814
zyklische PDOs 939
zyklische Task 814
Zyklische Übertragung 561
zyklischen Betrieb aktivieren 936
Zyklus
  Einzelzyklus ausführen 427
Zykluszeit 818
Zykluszeit CANbus 936
Zykluszeit EtherCAT Master 951
Zykluszeit-Verletzung 52

© 03/2011

Lenze Automation GmbH
Hans-Lenze-Str. 1
D-31855 Aerzen
Germany

+49 (0)51 54 / 82-0

+49 (0)51 54 / 82-28 00

Lenze@Lenze.de

www.Lenze.com

Service Lenze Service GmbH
Breslauer Straße 3
D-32699 Extertal
Germany

00 80 00 / 24 4 68 77 (24 h helpline)

+49 (0)51 54 / 82-11 12

Service@Lenze.de

SHPPLCD ▪ 13373835 ▪ DE 4.1 ▪ TD29

10 9 8 7 6 5 4 3 2 1